

Requirements and Design Specification Review

Command Support CSCI

Checkout and Launch Control Systems (CLCS)

84K00550

Approval:

Kirk Loughheed, Chief,
System Engineering and
Integration Division

Date

PREPARED BY:

REVISION HISTORY

REV	DESCRIPTION	DATE

LIST OF EFFECTIVE PAGES				
Dates of issue of change pages are:				
Page No.	A or D*	Issue or Change No.	CR No.	Effective Date**

Table of Contents

1. COMMAND SUPPORT CSCI	2
1.1 COMMAND SUPPORT INTRODUCTION	2
1.1.1 Command Support Overview	2
1.2 COMMAND SUPPORT SPECIFICATIONS	3
1.2.1 Command Support Common Ground Rules	3
1.2.2 Command Support Functional Requirements	4
1.2.3 Command Support Performance Requirements	4
2. COMMAND PROCESSOR CSC	5
2.1 COMMAND PROCESSOR INTRODUCTION	5
2.1.1 Command Processor Overview	5
2.1.2 Command Processor Operational Description	5
2.2 COMMAND PROCESSOR SPECIFICATIONS	6
2.2.1 Command Processor Ground Rules	6
2.2.2 Command Processor Functional Requirements	6
2.2.3 Command Processor Performance Requirements	7
2.2.4 Command Processor Interfaces Data Flow Diagram	7
2.3 COMMAND PROCESSOR DESIGN SPECIFICATION	7
2.3.1 Command Processor Detailed Data Flow	7
2.3.2 Command Processor External Interfaces	9
2.3.3 Command Processor Test Plan	11
3. COMMAND MANAGEMENT CSC	ERROR! BOOKMARK NOT DEFINED.
3.1 COMMAND MANAGEMENT INTRODUCTION	12
3.1.1 Command Management Overview	12
3.1.2 Command Management Operational Description	12
3.2 COMMAND MANAGEMENT SPECIFICATIONS	13
3.2.1 Command Management Ground Rules	13
3.2.2 Command Management Functional Requirements	13
3.2.3 Command Management Performance Requirements	14
3.2.4 Command Management Interfaces Data Flow Diagram	14
3.3 COMMAND MANAGEMENT DESIGN SPECIFICATION	16
3.3.1 Command Management Detailed Design	16
3.3.2 Command Management External Interfaces	17
3.3.3 Command Management Test Plan	18
4. COMMANDING INTERFACE FUNCTION	19
4.1 COMMANDING INTERFACE INTRODUCTION	19
4.1.1 Commanding Interface Overview	19
4.1.2 Commanding Interface Description	19
4.2 COMMANDING INTERFACE SPECIFICATIONS	19
4.2.1 Commanding Interface Ground Rules	19
4.2.2 Commanding Interface Functional Requirements	19
4.2.3 Commanding Interface Performance Requirements	21
4.2.4 Commanding Interface Interfaces Data Flow Diagram	21
4.3 COMMANDING INTERFACE DESIGN SPECIFICATION	22
4.3.1 Commanding Interface Detailed Design	22
4.3.2 Commanding Interface External Interfaces	23
4.3.3 Commanding Interface Test Plan	25
5. COMMAND SCRIPTER CSC	26

5.1 COMMAND SCRIPTER INTRODUCTION	26
5.1.1 Command Scripter Overview	26
5.1.2 Command Scripter Operational Description	26
5.2 COMMAND SCRIPTER SPECIFICATIONS	26
5.2.1 Command Scripter Ground Rules	26
5.2.2 Command Scripter Functional Requirements	26
5.2.3 Command Scripter Performance Requirements	27
5.2.4 Command Scripter Interfaces Data Flow Diagram	27
5.3 COMMAND SCRIPTER DESIGN SPECIFICATION	27
5.3.1 Command Scripter Class Diagrams	28
5.3.2 Command Scripter External Interfaces	28
5.3.3 Command Scripter Input Formats	30
5.3.4 Command Scripter Recorded Data	30
5.3.5 Command Scripter Printer Formats	30
5.3.6 Command Scripter Interprocess Communications	30
5.3.7 Command Scripter External Interface Calls	30
5.3.8 Command Scripter Table Formats	30
5.3.9 Command Scripter Test Plan	30
6. AUTHENTICATION SERVICE CSC	31
6.1 AUTHENTICATION SERVICE INTRODUCTION	31
6.1.1 Authentication Service Overview	31
6.1.2 Authentication Service Operational Description	32
6.2 AUTHENTICATION SERVICE SPECIFICATIONS	32
6.2.1 Authentication Service Ground Rules	32
6.2.2 Authentication Service Functional Requirements	32
6.2.3 Authentication Service Performance Requirements	33
6.2.4 Authentication Service Interfaces Data Flow Diagram	33
6.3 AUTHENTICATION SERVICE DESIGN SPECIFICATION	33
6.3.1 Authentication Service Detailed Design	34
6.3.2 Authentication Service External Interfaces	36
6.3.3 Authentication Service Test Plan	39

Table of Figures

Figure 1-1 Command Flow Overview	2
Figure 1-2 Command Support Overview with CORBA	3
Figure 2-1 Command Processor Overview	5
Figure 3-1 Command Management Overview	Error! Bookmark not defined.
Figure 3-2 Command Management External Interfaces	15
Figure 3-3 Command Management Detailed Data Flow	16
Figure 3-4 Command Management Class Diagram	17
Figure 4-1 Commanding Interface External Interface Data Flow Diagram	21
Figure 4-2 Commanding Interface Detailed Data Flow Diagram	22
Figure 4-3 Commanding Interface top level class diagram	23
Figure 6-1 Authentication Service Macroscopic Data Flow	32
Figure 6-2 Authentication Process	34
Figure 6-3 Authentication Service Detailed Data Flow	35
Figure 6-4 Authentication Service Class Diagram	36

REQUIREMENTS AND DESIGN SPECIFICATION

REVIEW COMMAND SUPPORT CSCI

CHECKOUT AND LAUNCH CONTROL SYSTEMS (CLCS)

1. COMMAND SUPPORT CSCI

1.1 COMMAND SUPPORT INTRODUCTION

The Command Support CSCI provides users and user applications (EIMs and TASs) the ability to command the RTPS. The Command Support CSCI is detailed in the following sections:

Section 2 describes Command Processor, a command line user interface.

Section 3 describes Command Management, manages command transactions.

Section 5 describes Commanding Interface, the command API.

Section 6 describes Command Scripter, a command text file interface.

Section 4 describes Authentication Service, the authentication API.

1.1.1 Command Support Overview

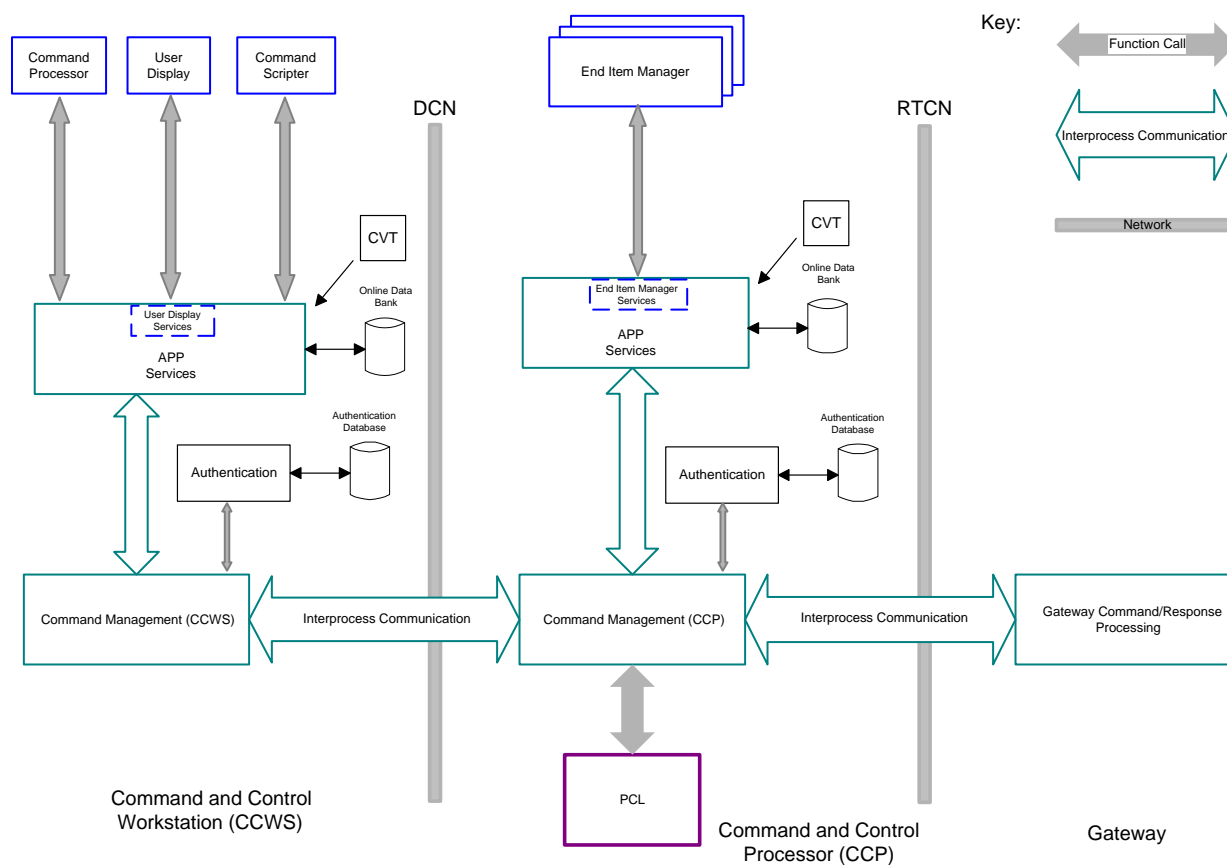


Figure 1-1 Command Flow Overview

Command Support facilitates the user and user applications in commanding the RTPS in the Command and Control Workstation (CCWS) and Command and Control Processor (CCP). There are three flows of commanding. The first is through the Command Processor or Command Scripter user interfaces on the CCWS. The second is through a GUI user application on the CCWS. The third is through a user application (EIM or TAS) executing in the CCP.

In the first command flow the Command Processor user interface accepts text input. The text is processed to form a command and then Application Services is called to create and issue the command through the Command Interface. Commands are received by the CCWS resident Command Management CSC. Command Management validates the command authentication and forwards the command to the CCP. A CCP resident Command Management CSC receives commands from CCWSs, performs authentication and FD PCL validation, and forwards the commands to the Gateways. Gateway Command Responses are sent back to the CCP Command Management and are then sent to the appropriate source.

In the second command flow, GUI applications command through “Hot Spots” on their screen. These applications interface with Application Services and the Commanding Interface to issue commands to the CCWS resident Command Management. From this point, command flow is similar to the Command Processor command flow.

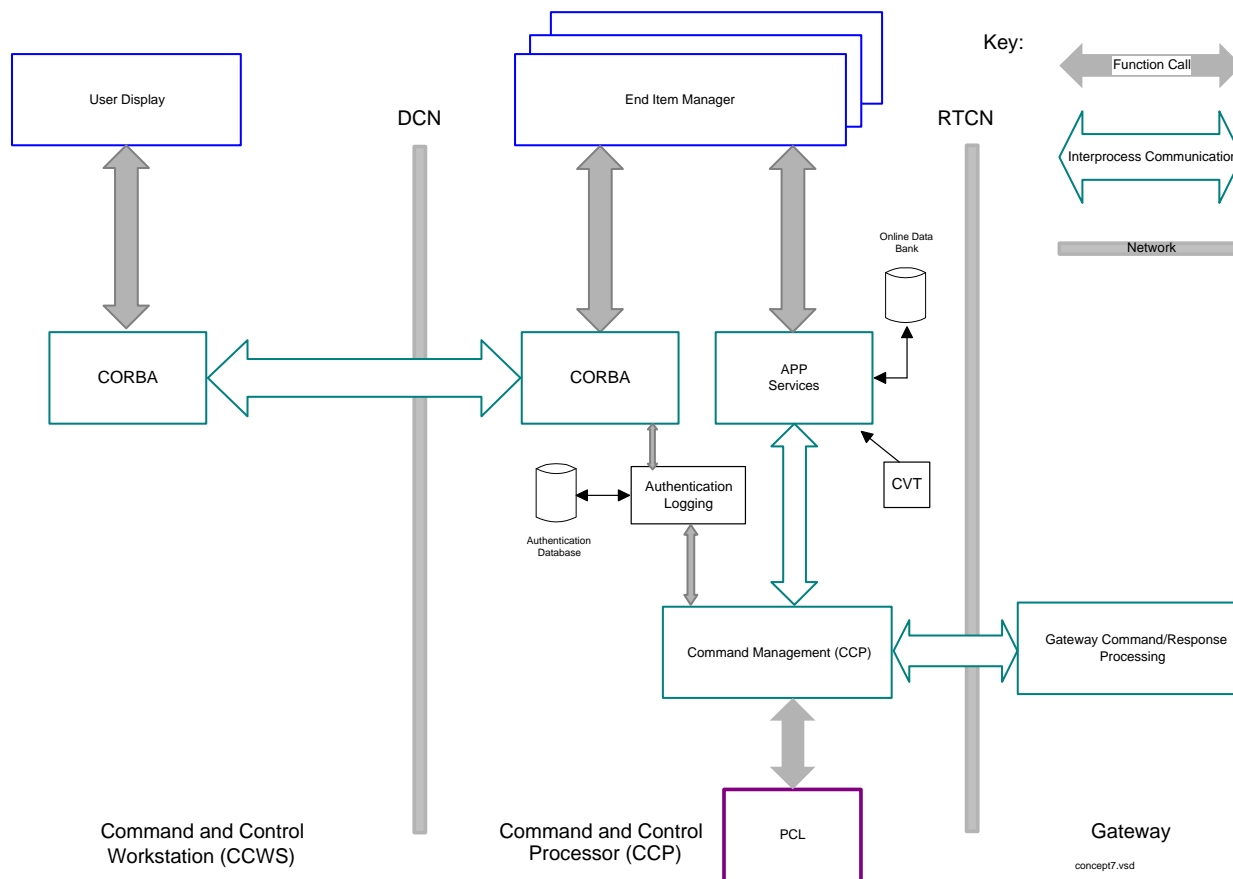


Figure 1-2 Command Support Overview with CORBA

The third command flow from a user application differs from the previous two flows in that commands originate from user applications executing on the CCP. The user applications are commanded through the CORBA interface. This interface calls the Authentication Interface for command Authentication. The user applications interface with Application Services and Commanding Interface to create objects and issue commands. Commands are sent directly to Command Management on the CCP where the command is authenticated for the application. Commands are then PCL checked before being routed to the appropriate Gateway. Gateway Command Responses are sent back to the CCP Command Management and are then sent to the appropriate source.

1.2 COMMAND SUPPORT SPECIFICATIONS

1.2.1 Command Support Common Ground Rules

N/A

1.2.2 Command Support Functional Requirements

N/A

1.2.3 Command Support Performance Requirements

See Command Support CSCs for performance requirements.

2. COMMAND PROCESSOR CSC

2.1 COMMAND PROCESSOR INTRODUCTION

2.1.1 Command Processor Overview

The Command Processor CSC resides in the CCWS and provides the capability for the CCWS user to issue Command Requests to end items attached to the RTPS. The Command Processor receives CCP Command Responses for each Command Request and displays an appropriate message on the Command Processor screen. The Command Processor issues a message to System Message Services for specific CCP Command Responses.

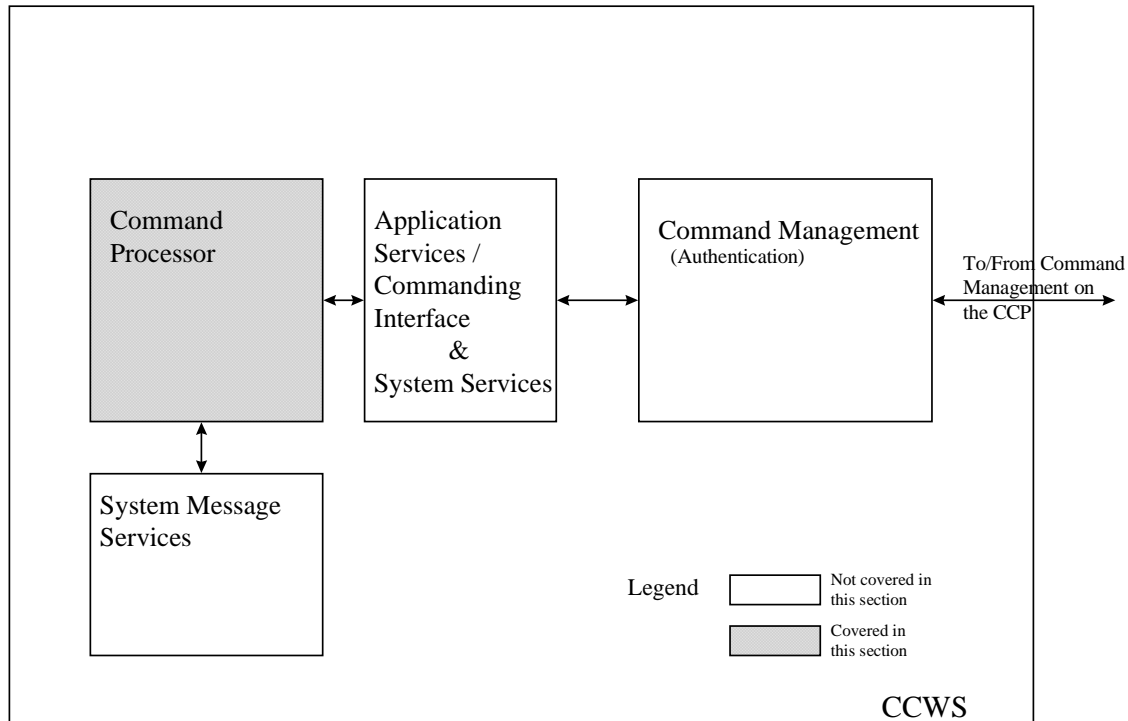


Figure 2-1 Command Processor Overview

2.1.2 Command Processor Operational Description

The Command Processor provides a means to enter Command Requests to end items attached to the RTPS by a user at a CCWS.

A Command Processor Command Request is made by typing a text string into a field on a CCWS User display screen. The Command Processor parses the text string to create a Command Request. Error messages are displayed

indicating syntax or commanding errors. The Command Processor validates whether the Command Request is for an existing FD and is a supported command. A Command Request will be sent if the entry is processed without error and accepted by Two Step Processing.

The Command Processor receives status about Command Requests from several stages in the command route. These messages consist of Two Step Command Processing error/status messages and CCP Command Responses. These messages/responses are displayed on the Command Processor screen and are sent to System Message Services.

2.2 COMMAND PROCESSOR SPECIFICATIONS

2.2.1 Command Processor Ground Rules

- User Display Services will be required to support the SL-GMS Command Processor to:
 - A) provide a modeless pop-up window for 2 step command options.
 - B) provide the user the ability to cancel a window.
- Application Services will be required to support the Command Processor to provide data to validate command syntax.
- System Services Interprocess Communications will be required to support the Command Processor CSC communications with the Command Management CSC.
- The Command Processor will output error messages to System Message Services.
- System Message Services shall provide an API that will return a formatted text string for the Command Processor to display.
- Command Processor shall be initialized by Control Navigation System.

2.2.2 Command Processor Functional Requirements

1. The Command Processor shall provide a user interface for command input.
2. The Command Processor interface shall accept keyboard input.
3. Commands shall be case insensitive.
4. A blank delimiter is equal to one or more blank characters.
5. A comma delimiter is equal to exactly one comma.
6. The Command Processor shall issue command requests.
7. The Command Processor shall accept command syntax as specified in Appendix A.
8. The Command Processor shall receive Command Responses from the CCP concerning each Command Request. The responses shall be routed to the User's Screen and System Message Services according to the table below.

	CCP Command Response	Command Processor Message	Additional Messages from Command Processor
a	Authentication Problem	Error Message	No Message
b	Command Time-out	Error Message	No Message
c	Command issued	No Message	No Message
d	Command completed	Status Message	No Message
e	Command Error (non-HIM error)	Error Message	Message to System Message Services
f	Command Error (HIM error)	Error Message	No Message

9. The Command Processor shall restrict invalid Command Requests
 - 9.1 The Command Processor shall reject invalid FD names.
 - 9.2 The Command Processor shall reject values out-of-range for an intended FD.
 - 9.3 The Command Processor shall reject invalid value types for an intended FD.

10. The Command Processor shall support two-step Command Requests (arm first, then execute).
11. The Command Processor shall interface with System Message Services API to retrieve command status text messages to be displayed on the Command Processor Interface.
12. The system shall support executing a manual command in less than one second from human execution to RTPS interface output.

2.2.3 Command Processor Performance Requirements

SLS [2.2.2.1.14 The system shall support executing a manual command in less than one second from human execution to RTPS interface output.](#)

2.2.4 Command Processor Interfaces Data Flow Diagram

Figure 2-1 shows the data flow for the interface to the Command Processor. The command is entered from the keyboard to the Command Processor GUI. The command string is then passed to the parser/builder which returns a Command object containing an error code indicating if the command was syntactically correct. If applicable to the command, a Two Step Window is displayed requesting confirmation before the command is sent to Command Management on the CCWS.

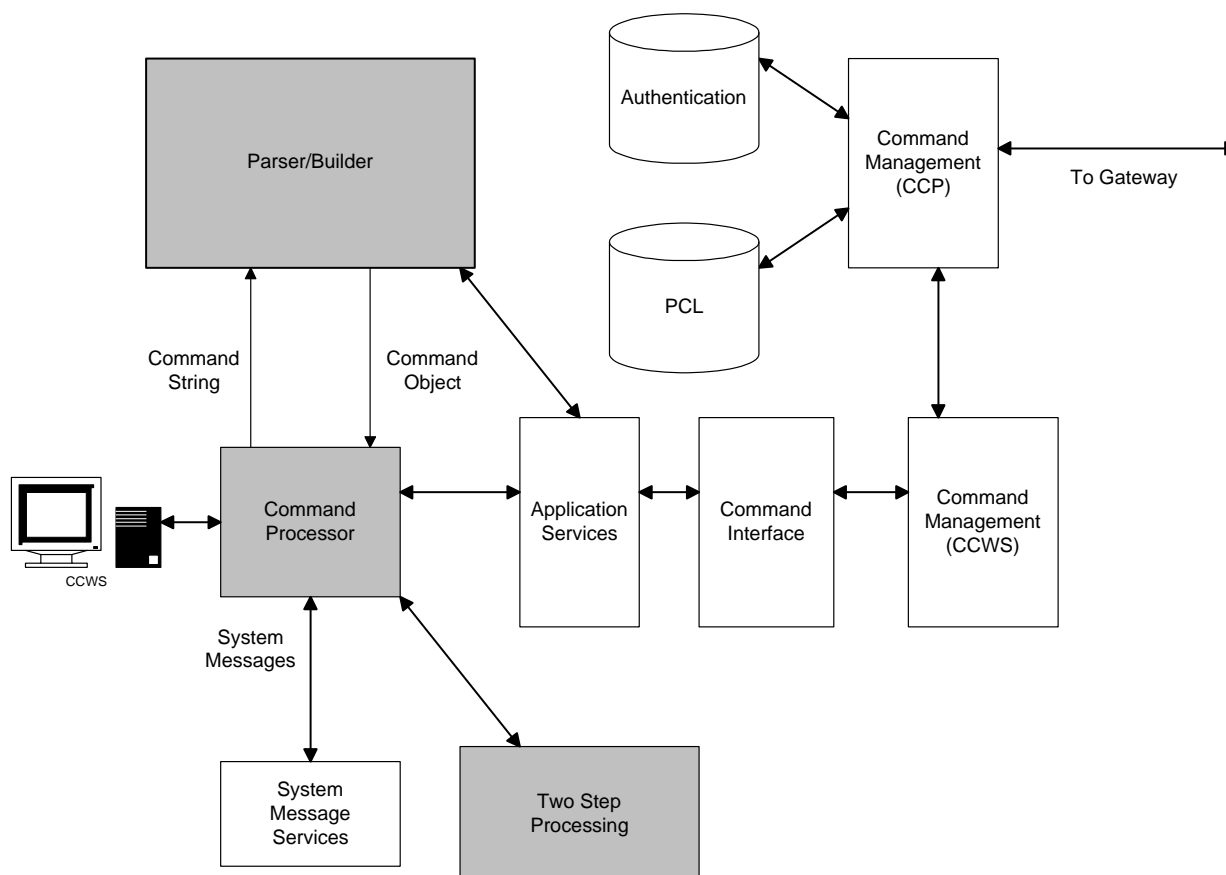


Figure 2-1 Command Processor Interfaces Data Flow Diagram

2.3 COMMAND PROCESSOR DESIGN SPECIFICATION

2.3.1 Command Processor Detailed Data Flow

2.3.1.1 Command Processor Design

The Command Processor Detailed Data Flow is shown in Figure 2-2. A user inputs a command to the command line on the Command Processor GUI. After hitting either the Send button on the GUI or the Enter key on the keyboard, the Command Processor retrieves the command from the command line as a character string. This string is passed to a Parser object, which breaks the string up into tokens delineated by either white space or a single comma. Once the command type is determined, Application Services is called to create either a GatewayCommanding or an FDCCommanding object. The Parser checks the command for proper syntax (as defined in Appendix A) and also verifies that the command's parameters form a valid command (ex. only a discrete stimulus FD is used in a Set command or that an Apply command is using values within the allowable range for the particular analog FD).

The Command object, containing an error code, is passed back to the Command Processor. The Command Processor uses the error code to determine if the command contained an error, and if so, what the error was. If the command does contain an error, the Command Processor discards the command, displays an appropriate message, and issues the appropriate System Message. If the command contained no errors, a Two Step Window is displayed in the case of two step commands or the command is issued through the FDCCommanding or GatewayCommanding object.

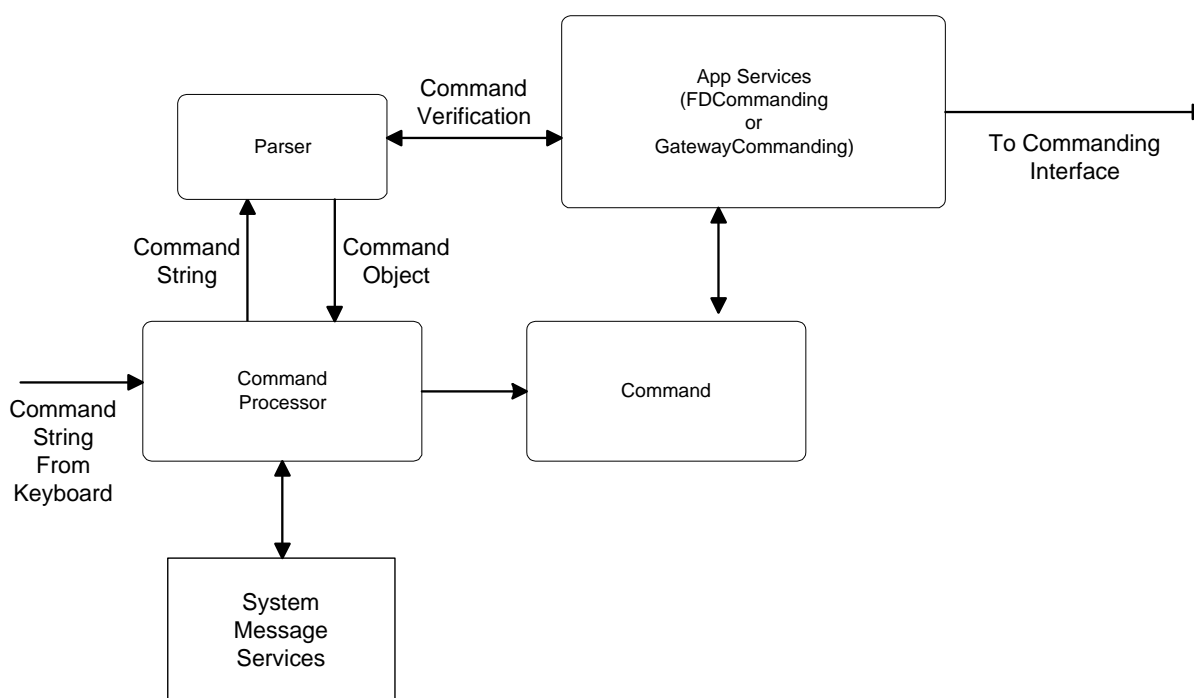


Figure 2-2 Command Processor Detailed Data Flow

2.3.1.2 Command Processor Class Diagrams

The Command Processor Class Diagrams are shown in Figure 2-3. The Command Processor class is the primary class of Command Processor. It contains the methods for controlling where a command is sent in various stages of processing. The Parser class does all processing of the command string, including syntax and error checking. The TwoStepWindow class is created to arm a two step command. It displays a GUI which lists the command and its options and gives the user the choice to cancel or send the command. The Command class is used to pass back information on a parsed command to the Command Processor. It includes a status code and an instance of a commanding object.

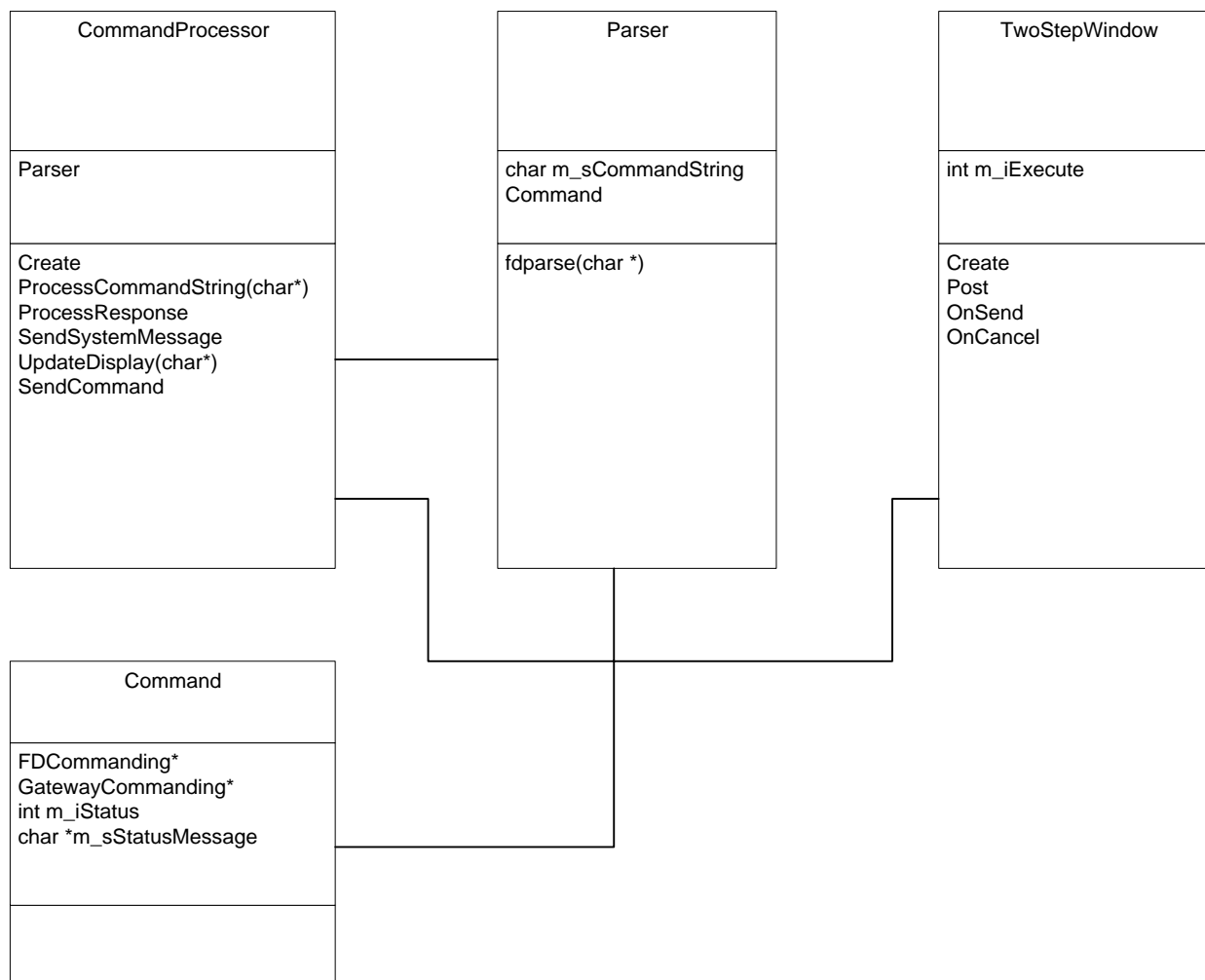


Figure 2-3 Command Processor Class Diagrams

2.3.2 Command Processor External Interfaces

2.3.2.1 Command Processor Message Formats

The Command Processor will issue the following responses* to the Command Processor window when processing a string input or a Command Response. Refer to the corresponding message number in the System Message Services catalogue for description and format of message.

SMS Message Number	SMS Message Mnemonic
41	CMD_CMP_APPLY_FAILED_INVALID_VALUE
42	CMD_CMP_APPLY_SUCCESSFUL
43	CMD_CMP_SET_FAILED_INVALID_STATE
44	CMD_CMP_SET_SUCCESSFUL
45	CMD_CMP_INVALID_COMMAND
46	CMD_CMP_INVALID_FD
47	CMD_CMP_CANCELED_IN_TWOSTEP
48	CMD_CMP_COMMAND_FAILED_INVALID_FD_TYPE

49	CMD_CMP_COMMAND_CURRENTLY_IN_TWOSTEP
53	CMD_CMP_APPLY_FAILED_OUT_OF_RANGE
54	CMD_CMP_SET_FAILED_INVALID_ARGUMENT

* More messages will be added to this list as commands become available.

2.3.2.2 Command Processor Display Formats

The Command Processor display is shown in Figure 2-4. It contains a text entry line where the commands are entered, and a scrolling list box which displays the entered command and command responses.

There are three buttons: *Clear Selected*, *Clear All*, and *Freeze* which control the display behavior of the scrolling list region. *Clear Selected* removes any selected lines from the scrolling list. *Clear All* removes all lines from the scrolling list. *Freeze* operates as a toggle button to either suspend or recommence scrolling of the list. All commands and responses issued or received while scrolling is suspended will be maintained internally in the list. However, they will not show on the display until the *Freeze* button has been toggled off.

The *Enter* and *Clear* buttons control the text entry line. *Enter* can be pressed instead of pressing the Return key when entering a command on the command line. *Clear* will simply clear off the command line without attempting to send the command.

The *Retrieve* button is used to retrieve a previously typed in command from the scrolling list region. A command is first selected from the scrolling list, then *Retrieve* is pressed. The command will be placed in the command line, where either the Return key or the *Enter* button can be used to send the command.

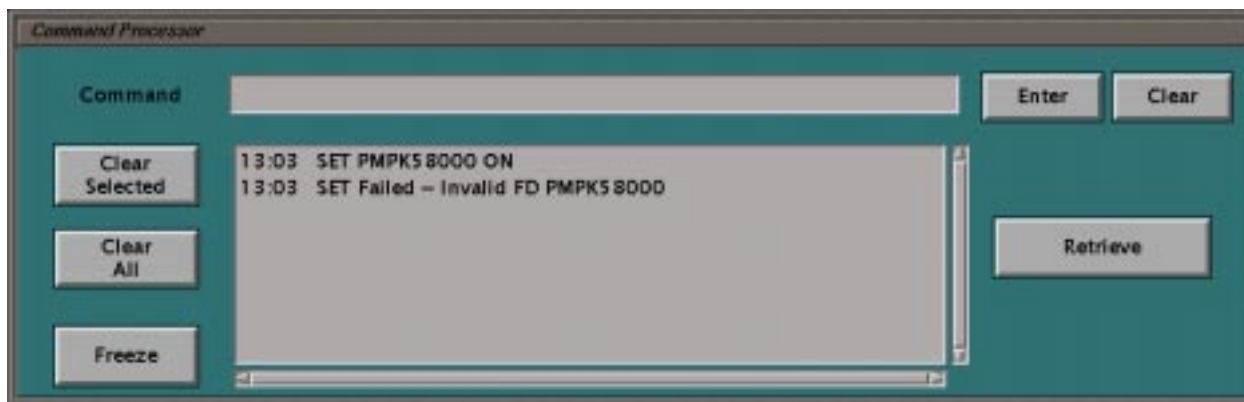


Figure 2-4

The Two Step Window display is shown in Figure 2-5. It contains a text area which displays information about the command being armed including the command name, and what parameters were specified. Two buttons are provided to either send or cancel a command request.

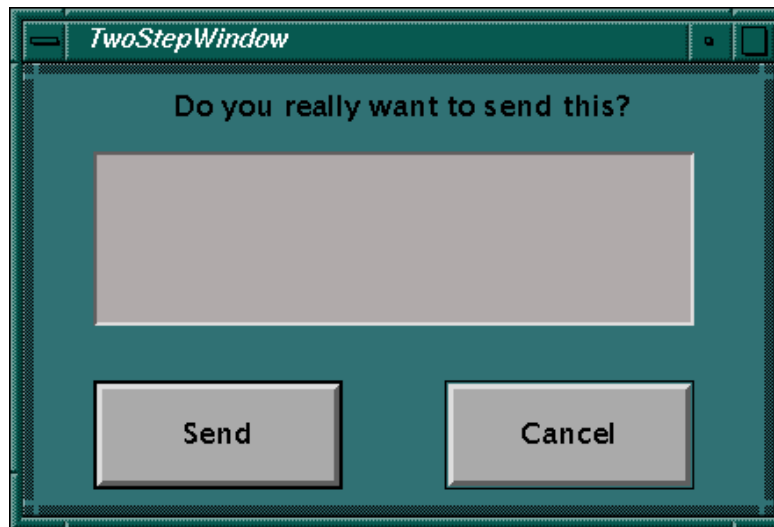


Figure 2-5

2.3.2.3 Command Processor Input Formats

Command Processor input formats consists of commands adhering to the syntax rules given in [Appendix A CLCS Commands](#).

2.3.2.4 Command Processor Recorded Data

N/A

2.3.2.5 Command Processor Printer Formats

N/A

2.3.2.6 Command Processor Interprocess Communications (C-to-C Communications)

System Services will be used to support interprocess communications between Command Processor and Command Management.

2.3.2.7 Command Processor External Interface Calls (e.g., API Calling Formats)

Command Processor will make external calls to System Messages and Application Services. System Messages will be issued by the Command Processor to indicate any of the errors given in section 2.3.2.1. Application Services will be used to provide data for validation of commands..

2.3.2.8 Command Processor Table Formats

N/A

2.3.3 Command Processor Test Plan

1. Verify the Command Processor accepts command syntax as specified in Appendix A.
2. Verify the Command Processor returns an appropriate error code on entering erroneous commands.
3. Verify the Command Processor issues command requests.
4. Verify the Command Processor supports two-step Command Requests (arm first, then execute) for two step commands.
5. Verify the Command Processor interfaces with System Message Services API to issue System Messages.
6. Verify the Command Processor interfaces with System Message Services API to retrieve command status text messages to be displayed on the Command Processor Interface.

3. COMMAND MANAGEMENT CSC

3.1 COMMAND MANAGEMENT INTRODUCTION

3.1.1 Command Management Overview

Command Management is a service that provides Transaction Management of Command Requests and Responses as they are transported through the RTPS System. Commands are received at Command Management via the Commanding Interface Function, and are sent back to that interface when the command has been completed. The Command Management process also assures that Command Responses are received from their destination in a timely manner, or a time out message will be sent to the Commanding Interface that initiated the Command Request.

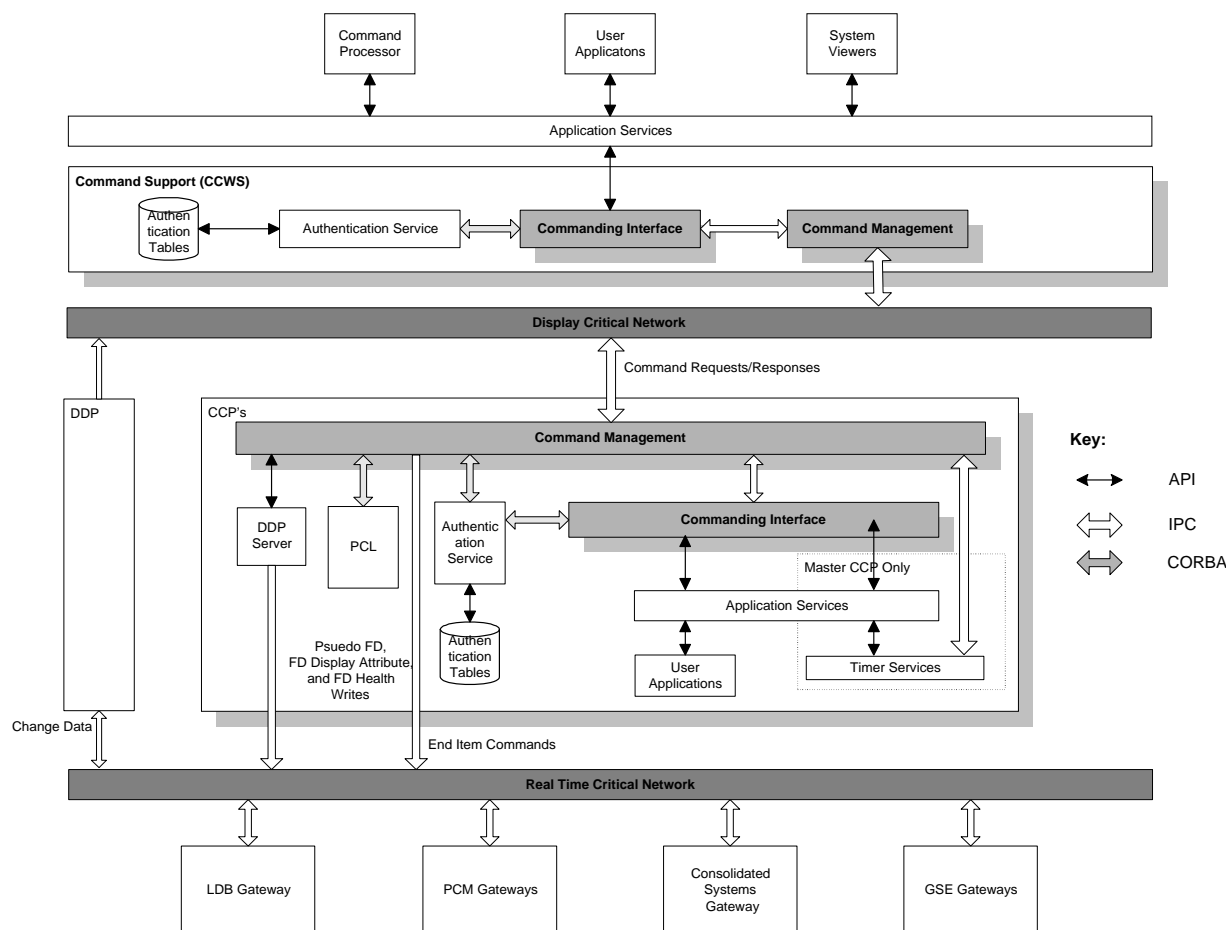


Figure 3-1 Command Management Overview

3.1.2 Command Management Operational Description

Command Management has two operation modes depending upon its platform location: CCPs or CCWS.

In general, Command Management receives Command Requests, performs authentication and stores information about the Command Requests into a transaction table. The Command Requests are then forwarded to their

destination, and the transaction table is monitored in case the Command Request should time out. Command Responses received from the destination(s) are matched with their entry in the transaction table and forwarded back to the requesting process. If a Command Request should time out, a Command Response will be generated with the appropriate error code and a system message will be sent to System Message Services.

Command Management on the CCWS calls the Authentication Service CSC to verify that the CCWS user application has the authority to issue the command. Command Management on the CCP performs two types of authentication depending on the source of the message. If the message originates from outside the Command Management's CCP, then Test Set authentication is performed. If the message originates from Command Management's CCP, then Command Management calls the Authentication Interface to verify that the CCWS application has the authority to issue the command.

Command Management on the CCP calls Data Distribution APIs to process Pseudo FD commands, change Data Health commands, and change Display Attributes commands.

3.2 COMMAND MANAGEMENT SPECIFICATIONS

3.2.1 Command Management Ground Rules

- Command Management processes will be started and terminated by the System Control CSCI.
- Command Management will be initialized on the CCWS and CCP platforms.
- Command Management, when initialized, will configure itself to perform certain levels of command authentication, depending on which platform it resides on.
- System Services Interprocess Communications will be required to support the Command Management to:
 - A) provide communications between system applications residing on the same platform.
 - B) provide communications between system applications residing on different platforms.
- Command Responses to Command Management must include a Packet Payload Header for each response, formatted in accordance with the Real Time Processing System Packet Payload ICD, Document Number 84K00351.
- Command Management will output error and status messages to System Message Services.
- The PCL CSC shall provide a PCL API.
- Data Distribution shall provide an API to update individual FD values.
- Data Distribution shall provide an API to update a FD's Display Attribute values.
- Data Health shall provide an API to update a FD's health values.
- Command Management will set the initial time out values as specified in the following table.

Command Management TimeOut Table	
Location of CM	TimeOut
CCWS	120 msec
CCP	50 msec

3.2.2 Command Management Functional Requirements

- 1 Command Management shall initialize itself and wait for Command Requests and Command Responses when Command Management is started by the System Control CSCI.
- 2 Command Management, on the CCWS platform, shall receive Command Requests from the Commanding Interface Function.
- 3 Command Management, on the CCP platform, shall receive CCWS Command Request messages formatted in accordance with the Real Time Processing System Packet Payload ICD.
- 4 Command Management, on the CCP platform, shall verify (Authenticate) that the CCWS Command Request is issued from a configured CCWS that is part of the Test Set.

- 5 Command Management shall track Command Requests for response matching for commands that require a Command Response when directed by the RTPS Packet Payload.
- 6 Command Management shall keep a TimeOut value for each Command Request sent.
 - 6.1 The Command Management shall set the TimeOut value to an initial value depending upon its host platform (CCWS or CCP)
 - 6.2 The Command Management shall issue an Error Response when a Command Request's TimeOut has been exceeded.
- 7 Command Management shall accept the following Command Responses:
 - 7.1 Error Response.
 - 7.2 Successful Response.
 - 7.3 Issued Response
- 8 Command Management shall accept Unsolicited Command Responses. Unsolicited Command Responses include those responses received for command requests that have timed out.
 - 8.1 The Command Management shall send a message to System Message Services for each Unsolicited Command Response received.
 - 8.2 The Command Management shall not attempt to route the Unsolicited Command Response back to the commanding source of the message.
- 9 Command Management shall match the Command Responses to the stored Command Requests in the transaction table.
- 10 Command Management shall forward Command Responses containing a status or error message to the originating process.

	Response Message (RM)	RM Returned	System Message
a	Invalid Source (Flow) for Command Request	Y	Y
b	GW Command (Error) Response	Y	N*
c	GW Command (Successful) Response	Y	N
d	CMD MGMT Generated Gateway Busy	Y	Y
e	CMD MGMT Generated Command Issued	Y	N
f	CMD MGMT Generated Command Time Out Error Command Response	Y	Y
g	GW Unsolicited Command Responses (error)	NY**	Y

* G/W will issue HIM error messages to System Message Services.

11. Command Management on the CCP shall route FD commands through the PCL interface.
12. Command Management shall update the timeout value for a command in the transaction table when processing an "issued response" Command Response.
13. Command Management shall interface with Data Distribution and Data Health APIs to publish Pseudo FD values, FD Health status and FD Display attributes, ~~and FD Constraint values.~~

3.2.3 Command Management Performance Requirements

- SLS 2.2.2.1.4 The system shall provide the capability for a single test (control) application to queue multiple LDB commands to support every LDB command opportunity.
- SLS 2.2.2.1.8 GSE command/response latency of a priority command, or of a non-priority command in an unloaded system, shall be less than 20 milliseconds from the time a test/control application issues the command until the response is received by the test application.
- SLS 2.2.2.1.9 GSE command/response latency of a non-priority command in a system supporting the "system maximum data bandwidth" and with 20 test applications executing in the same subsystem shall be less than 100 milliseconds from the time the test/control application issues the command until the response is received by the test application.

3.2.4 Command Management Interfaces Data Flow Diagram

11/19/97

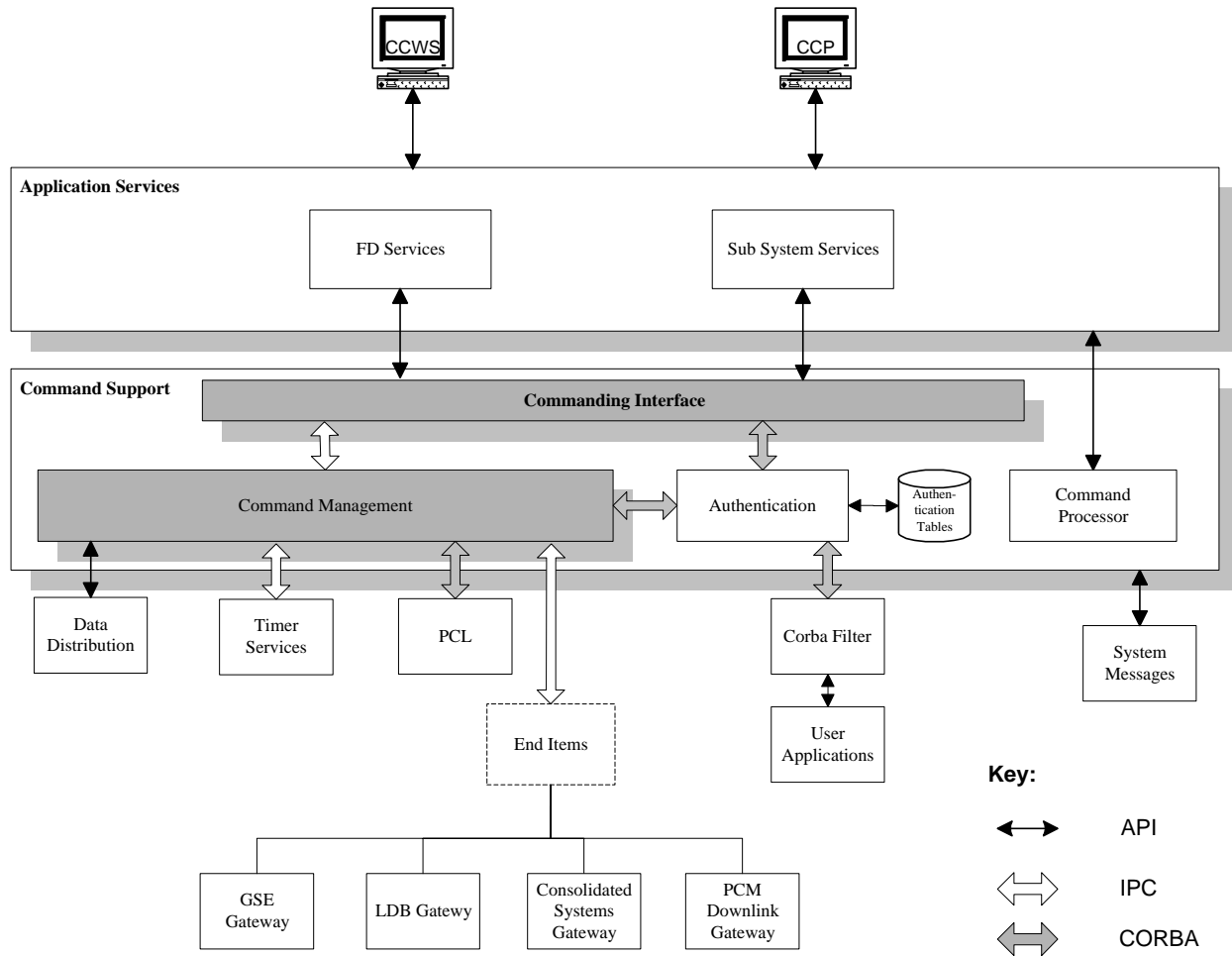


Figure 3-2 Command Management External Interfaces

3.3 COMMAND MANAGEMENT DESIGN SPECIFICATION

3.3.1 Command Management Detailed Design

3.3.1.1 Command Management Detailed Data Flow

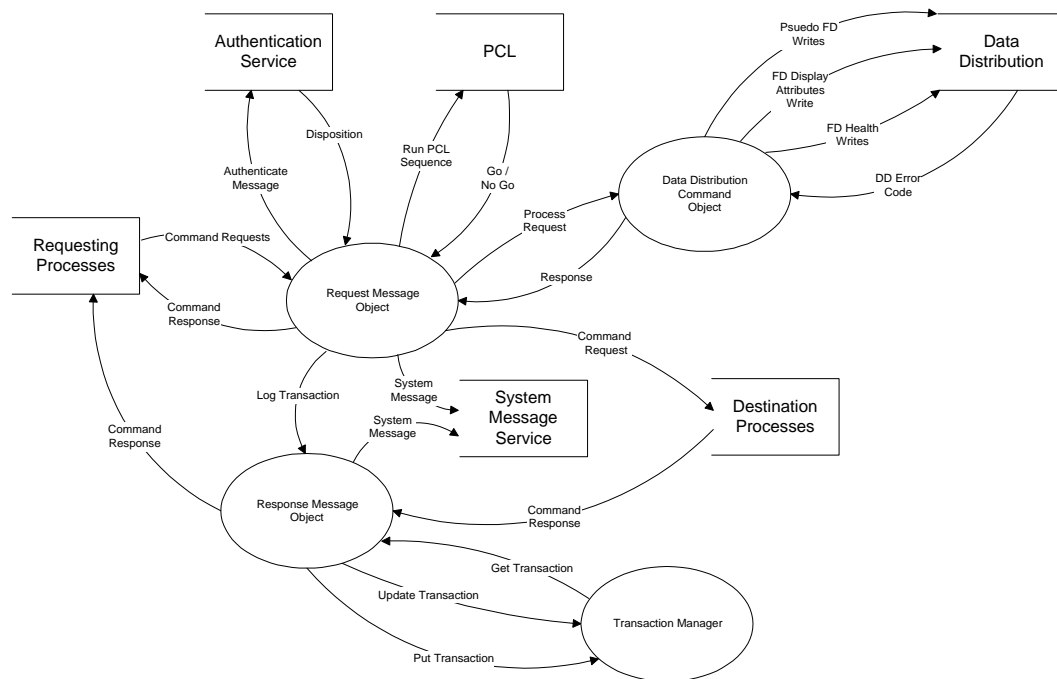


Figure 3-3 Command Management Detailed Data Flow

3.3.1.2 Command Management Class Diagrams

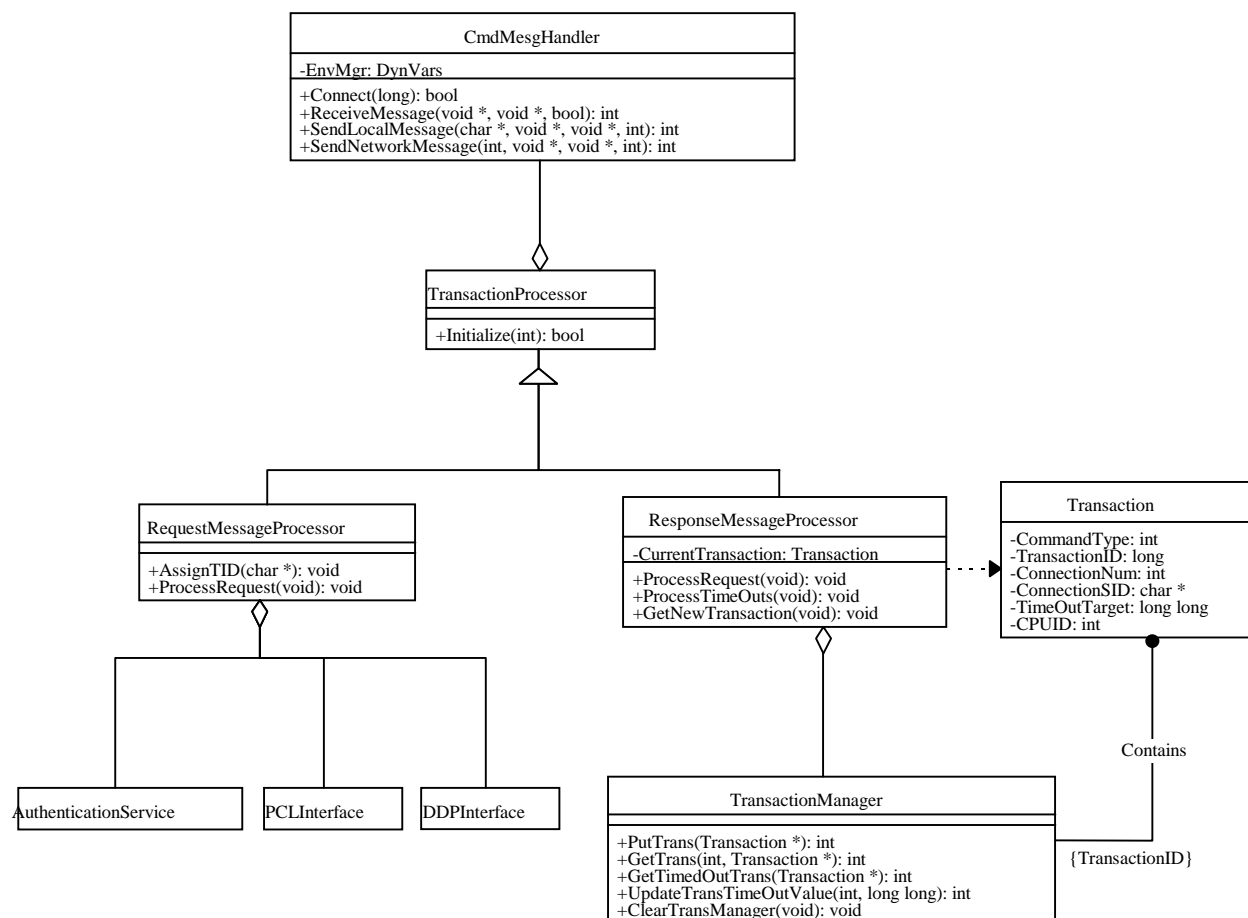


Figure 3-4 Command Management Class Diagram

3.3.2 Command Management External Interfaces

3.3.2.1 Command Management Message Formats

Command Management belongs to the Command Support CSCI Message Group.

The Command Management sends the following system messages using System Message Services:

Message Number	Message Text
101	CMD_CMM_IPC_CONNECT_FAILED
84	CMD_CMM_IPC_COMM_ERR
86	CMD_CMM_UNSOLICITED_RESPONSE_ERR
103	CMD_CMM_IPC_RECEIVE_FAILED
102	CMD_CMM_IPC_SEND_FAILED
85	CMD_CMM_TIMED_OUT_REQUEST_ERR
TBD	TBD

For a detailed description for each of the system messages listed in the previous table, see the System Message Catalogue located on the CLCS Development web page. As functionality is added to the Command Management Process, system messages will be added to the System Message Catalogue and captured in the above table.

3.3.2.2 Command Management Display Formats

N/A

3.3.2.3 Command Management Input Formats

N/A

3.3.2.4 Command Management Recorded Data

Command Management will set the logging flags in the RTPS Packet Header for all Command Requests and Responses. All information being sent across the DCN or RTCN Networks shall be recorded.

3.3.2.5 Command Management Printer Formats

N/A

3.3.2.6 Command Management Interprocess Communications

Command Management will send and receive IPC Messages according to the following Interface Description Documents listed below:

Document Number	Document Name
84K0XXX	GSE Gateway IDD
84K0XXX	LDB Gateway IDD
84K0XXX	PCM Downlink Gateway IDD
84K0XXX	Consolidated Systems Gateway IDD
84K0XXX	Timer Services IDD

3.3.2.7 Command Management External Interface Calls

Command Management will use APIs contained in the following documents:

Document Number	Document Name
TBD	Data Distribution API
TBD	IPC Services API
TBD	Platform Parameter Table API
TBD	Authentication Service IDD (CORBA)
TBD	PCL Services IDD (CORBA)

3.3.2.8 Command Management Table Formats

N/A

3.3.3 Command Management Test Plan

TBS

4. COMMANDING INTERFACE FUNCTION

4.1 COMMANDING INTERFACE INTRODUCTION

4.1.1 Commanding Interface Overview

The Commanding Interface Function consists of a set of System Level APIs used for sending commands through the RTPS System. The Commanding Interface APIs send Command Requests and Responses to a Command Management process which manages the status of the commands. Responses to the Command Requests are returned to the requesting applications when completed.

4.1.2 Commanding Interface Description

The purpose of the Commanding Interface Function is to provide a means of sending commands from User Applications to End Items or System Level Applications in the RTPS System, without needing specific knowledge of the underlying Systems Services. Command Requests and Responses are transported to a Command Management Process which monitors all transactions the commands encounter to and from their destinations. The Command Responses are returned from the Command Management Process with a completion code that represents the success or failure of the Command Request.

The Commanding Interface APIs are provided in an Object Oriented fashion so that commands are “logically grouped” according to the purpose of the command. For instance, commands that control specific FDs are provided in a FD Commanding Object, commands that control table maintenance on a Gateway are provided in a Gateway Commanding Object, and so on. The Commanding Interface APIs are accessed through the Application Services layer, which contain a set of APIs for User Applications.

4.2 COMMANDING INTERFACE SPECIFICATIONS

4.2.1 Commanding Interface Ground Rules

- Commands will be verified by the application for content/grammar prior to calling the Commanding Interface APIs.
- User/System Applications will access the Commanding Interface APIs through the Application Services CSCI Interface.
- The RTPS Packet shall provide a field to specify routing of LDB Commands to a priority queue on the LDB.
- *OLDB shall provide initial routing information to a CCP and Gateway for all FDs, including Pseudo FDs.*
- *OLDB shall provide routing information to a DDP for all Pseudo FDs.*

4.2.2 Commanding Interface Functional Requirements

- 1 The Commanding Interface shall provide a means to specify PCL Override on FD Commands.
- 2 The Commanding Interface shall build and route the following commands to the support Ground Support Equipment Gateway Commands.
 - 2.1 APPLY (Apply Analog)
 - 2.2 SET (Set Discrete)
 - 2.3 ISSUE (Issue Digital Pattern)
 - 2.4 R FD (Read GSE HIM Output FD)
 - 2.5 R EUC (Read EU Coefficients)
 - 2.6 A/I HT (Activate/Inhibit HIM Testing Command)
 - 2.7 A/I DA (Activate/Inhibit Data Acquisition)
 - 2.8 A/I PR (Activate/Inhibit Data Processing)
 - 2.9 A/I CMD (Activate/Inhibit Command Issuance)

11/19/97

- 2.10 A/I HI (Activate/Inhibit HIM Polling Command)
- 2.11 A/I ~~CPSI~~ (Activate/Inhibit Change Process)
- 2.12 C HA (Change Hardware Address)
- 2.13 C RA (Change Sample Rate)
- 2.14 C EUC (Change EU Coefficients)
- 2.15 S FD (Status FD)
- 3 The Commanding Interface shall build and route the following commands to the~~provide Initial~~ Launch Data Bus Gateway Commands.
 - 3.1 APPLY (Apply Analog)
 - 3.2 SET (Set Discrete)
 - 3.3 ISSUE (Issue Digital Pattern)
 - 3.4 CBTU MR (LDB MDM Master Reset)
 - 3.5 CBTU LBSR (LDB Load BITE Status Register)
 - 3.6 CBTU BT1 (LDB BITE Test 1)
 - 3.7 CBTU BT3 (LDB BITE Test 3)
 - 3.8 READ (Read Onboard Values)
 - 3.9 CBTU RBSR (LDB Read BITE Status Register)
 - 3.10 CBTU WRAP (LDB Return Received Command Word)
 - 3.11 CBTU BT2 (LDB BITE Test 2)
 - 3.12 CBTU BT4 (LDB BITE Test 4)
 - 3.13 A/I DA (Activate/Inhibit Data Acquisition)
 - 3.14 ISSUE DEU Commands:
 - 3.14.1 *OPS*
 - 3.14.2 *SPEC*
 - 3.14.3 *I/O RESET*
 - 3.14.4 *RESUME*
 - 3.14.5 *MSG RESET*
 - 3.14.6 *ACK*
 - 3.14.7 *FAULT SUMM*
 - 3.14.8 *EXEC*
 - 3.14.9 *GPC CRT*
 - 3.14.10 *SYS SUMM*
 - 3.14.11 *ITEM*
 - 3.14.12 *DEST*
 - 3.15 GPCC LDBC (LDB Control)
 - 3.16 TEXT (LDB Send DEU Text Message to GPC)
 - 3.17 GMEM Read
 - 3.17.1 *Contiguous GPC Read*
 - 3.17.2 *Random GPC Read*
 - 3.17.3 *DEU1 Read*
 - 3.17.4 *DEU2 Read*
 - 3.17.5 *DEU3 Read*
 - 3.17.6 *DEU4 Read*
 - 3.18 GMEM Write
 - 3.18.1 *Set Bits (Protected/Unprotected)*
 - 3.18.2 *Reset Bits (Protected/Unprotected)*
 - 3.18.3 *Random Write (Protected/Unprotected)*
 - 3.18.4 *Contiguous Write (Protected/Unprotected)*
 - 3.19 S FD (Status FD)
- 4 The Commanding Interface shall provide a means of routing commands to a Priority Queue for LDB Commands.
- 5 The Commanding Interface shall build and route the following commands to the~~provide Initial~~ PCM Gateway Commands.
 - 5.1 R EUC (Read EU Coefficients)
 - 5.2 A/I DA (Activate/Inhibit Data Acquisition)
 - 5.3 A/I PR (Activate/Inhibit Data Processing)

- 5.4 A/I SI (Activate/Inhibit Significant Change Check)
- 5.5 A/I FL (Activate/Inhibit Frame Logging)
- 5.6 C EUC (Change EU Coefficients)
- 5.7 C PSB (Change Sync Bits in Error Count)
- 5.8 PCMS (Change/Select PCM)
- 5.9 S FD (Status FD)
- 6 The Commanding Interface shall format Command Requests in accordance to the RTPS Packet Payload ICD, Document Number 84K00351.
- 7 The Commanding Interface shall accept Command Responses formatted in accordance to the RTPS Packet Payload ICD, Document Number 84K00351.
- 8 ~~The Commanding Interface shall provide Initial Constraint Management Commands.~~
- 8 The Commanding Interface shall build and route the following commands to Data Distribution.
 - 1. Change Display Attributes for a given FD.
 - 2. Change Failure and Warning Reason Codes for a given FD.
- 9 The Commanding Interface shall ~~build and route the following commands to the~~provide Initial Timer Services Process on the Master CCP Commands.
 - 9.1 Set CDT/MET to a value
 - 9.2 Start CDT/MET
 - 9.3 Start CDT/MET at a specified GMT
 - 9.4 Hold CDT/MET
 - 9.5 Hold CDT/MET at a specified CDT/GMT
 - 9.6 Cancel pending CDT/MET command

4.2.3 Commanding Interface Performance Requirements

N/A

4.2.4 Commanding Interface Interfaces Data Flow Diagram

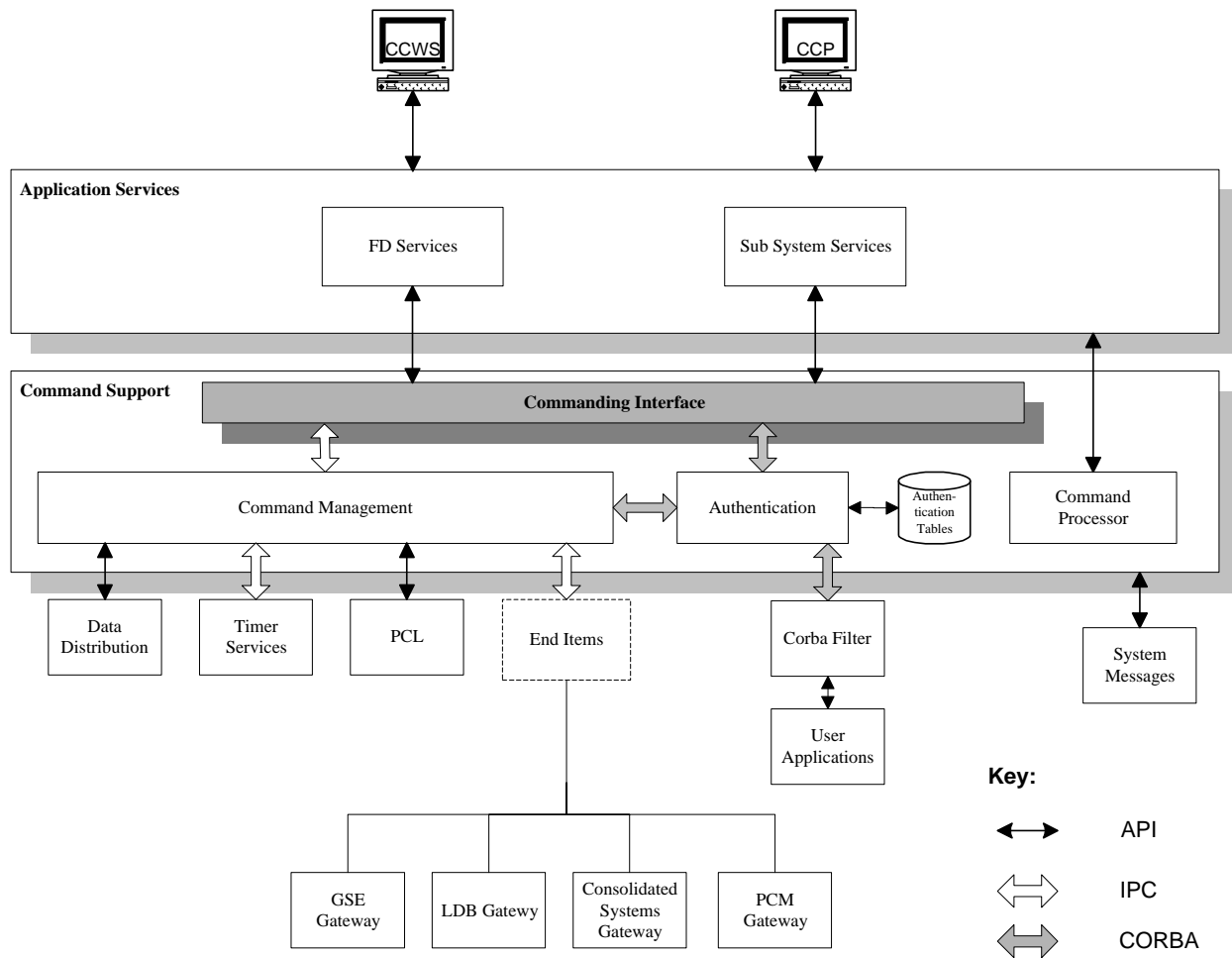


Figure 4-1 Commanding Interface External Interface Data Flow Diagram

4.3 COMMANDING INTERFACE DESIGN SPECIFICATION

4.3.1 Commanding Interface Detailed Design

4.3.1.1 Commanding Interface Detailed Data Flow

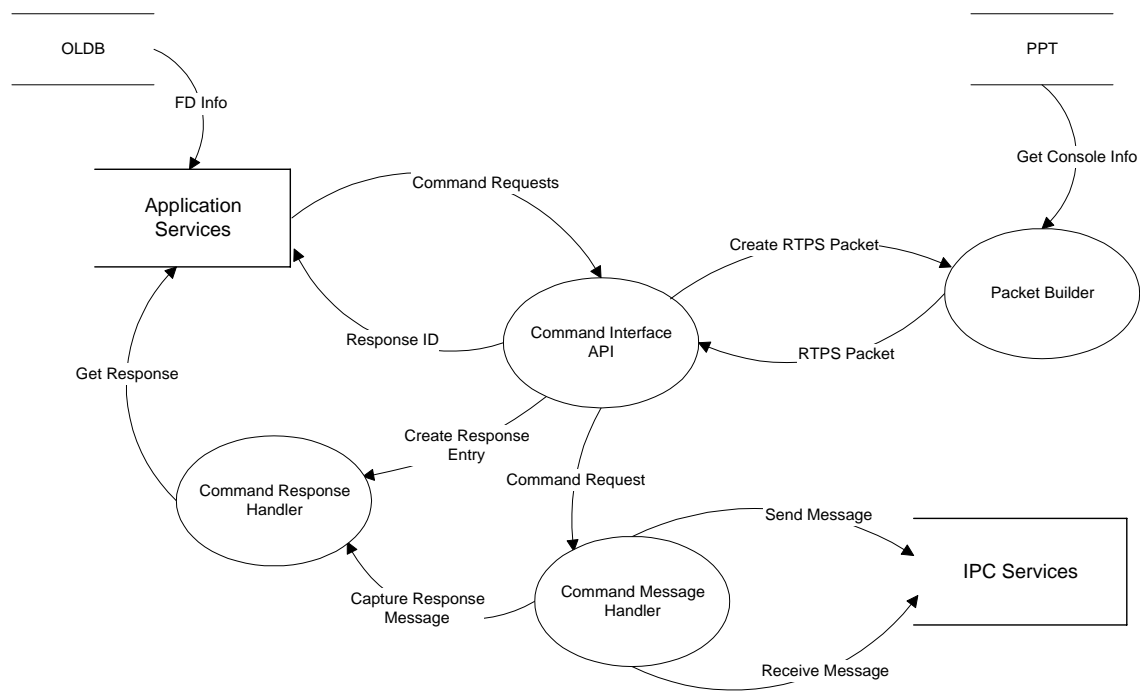


Figure 4-2 Commanding Interface Detailed Data Flow Diagram

4.3.1.2 Commanding Interface Class Diagrams

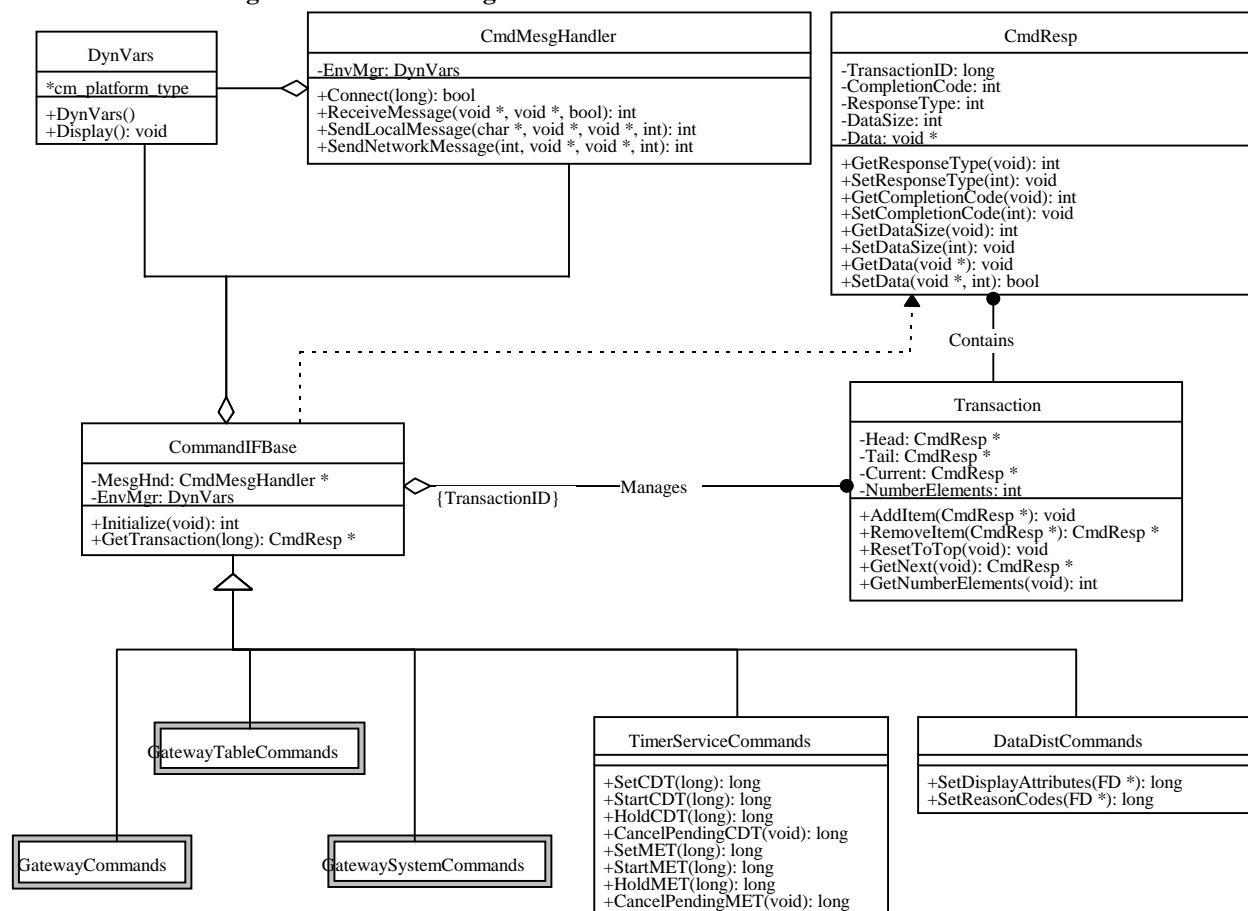


Figure 4-3 Commanding Interface top level class diagram

4.3.2 Commanding Interface External Interfaces

4.3.2.1 Commanding Interface Message Formats

The Commanding Interface sends the following system messages using System Message Services:

Message Number	Message Text
88	CMD_FDC_INVALID_PLATFORM_TYPE
101	CMD_CMM_IPC_CONNECT_FAILED
TBD	TBD

For a detailed description of each of the system messages listed in the previous table, see the System Message Catalogue located on the CLCS Development Web Page. As functionality is added to the Commanding Interface, system messages will be added to the System Message Catalogue and captured in the above table.

4.3.2.2 Commanding Interface Display Formats

N/A

4.3.2.3 Commanding Interface Input Formats

N/A

4.3.2.4 Commanding Interface Recorded Data

N/A

4.3.2.5 Commanding Interface Printer Formats

N/A

4.3.2.6 Commanding Interface Interprocess Communications

N/A

4.3.2.7 Commanding Interface External Interface Calls

The Commanding Interface will reference APIs from the following documents:

Document Number	Document Name
TBD	Authentication Service CSC API
TBD	IPC Services API
TBD	Platform Parameter Table API
TBD	FD Services API

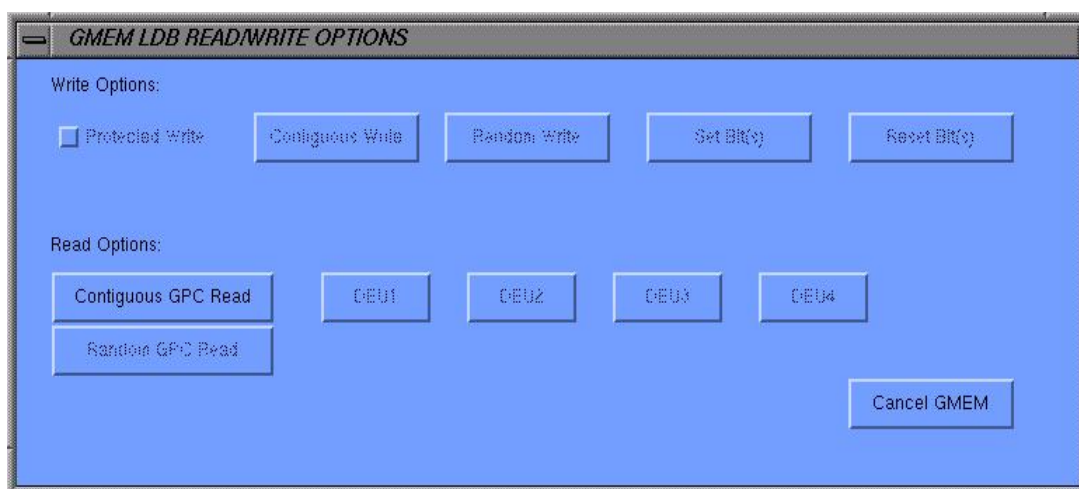
4.3.2.8 Commanding Interface Table Formats

N/A

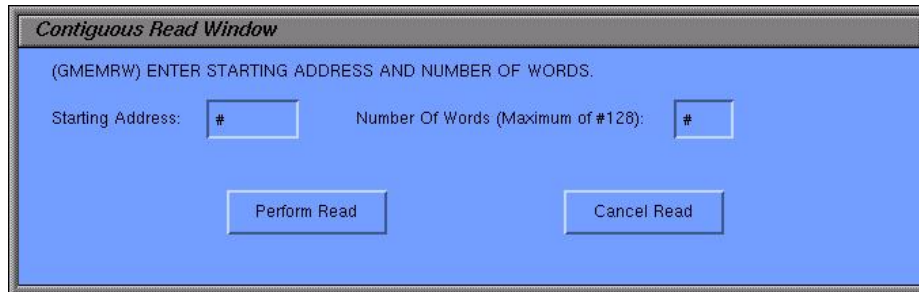
4.3.3 Commanding Interface Test Plan

The Commanding Interface will utilize the End to End Test Plan provided in the Command Management CSC Section of this document.

The Commanding Interface Function will use the following prototype GUI interfaces to implement some of the test procedures. These GUI Interfaces will be developed in conjunction with the System Viewer CSCI, and Human Factors.



GMEM Read/Write Main Window Prototype



The image shows a software window titled "Contiguous Read Window". Inside the window, the text "(GMEMRW) ENTER STARTING ADDRESS AND NUMBER OF WORDS." is displayed. Below this text, there are two input fields. The first is labeled "Starting Address:" and contains a hash symbol "#". The second is labeled "Number Of Words (Maximum of #128):" and also contains a hash symbol "#". At the bottom of the window, there are two buttons: "Perform Read" on the left and "Cancel Read" on the right.

Contiguous GMEM Read Input Window Prototype

5. COMMAND SCRIPTER CSC

5.1 COMMAND SCRIPTER INTRODUCTION

5.1.1 Command Scripter Overview

The Command Scripter is a GUI application that will allow users to Enter, Edit and Execute text files containing simulated operator inputs. These files are used to store commands in a CPRO format, just as they would be entered from the command line.

5.1.2 Command Scripter Operational Description

The function of the Command Scripter is to read simulated operator inputs from a disk file and feed them into the CLCS System as if the operator were keying them in manually. A GUI interface will be provided to view the script files, and allow the operator to edit and add lines to the file as needed. The GUI interface will also allow the operator to execute the file using Control Buttons which can control the execution of the file, such as changing mode, changing time delays, and setting breakpoints.

The Command Scripter will also accept imbedded commands from the disk files. This special set of commands can control the execution of the file, much like the Control Buttons on the GUI interface.

5.2 COMMAND SCRIPTER SPECIFICATIONS

5.2.1 Command Scripter Ground Rules

- Command Scripts will be stored as ASCII text files.
- Commands supported by the Commanding Scripter CSC will consist of, not to exceed, commands listed in Appendix A.
- Mode Control Commands entered in the script files will be surrounded by the '<' and '>' characters. i.e. <DELAY 10>.

5.2.2 Command Scripter Functional Requirements

1. The Command Scripter CSC shall provide a GUI Interface for editing/executing script files.
2. The Command Scripter shall provide syntax checking on the script files.
3. The Command Scripter shall display any errors in the syntax when the file is loaded onto the GUI Interface.
4. The Command Scripter shall provide a manual input process for editing or entering commands.
5. The Command Scripter shall accept ASCII script files created from another source.
6. The Command Scripter shall provide the ability to Save a new or modified script file.
7. The Command Scripter shall provide an "Override" capability from the GUI Interface to change the mode of execution of the script file.
8. The Command Scripter shall accept a set of Mode Control Commands which can be embedded in the script file. These Mode Control Commands are:
 - 8.1 <DELAY X> or <DEL X> where X is the number of milliseconds to delay between executable lines.
 - 8.2 <FAST> or <FAS> to change the mode of execution, executing as fast as the system can handle.
 - 8.3 <SLOW> or <SLO> to change the mode of execution, pausing for 3 seconds between executable lines.
 - 8.4 <SINGLE STEP> or <SS> to change the mode of execution, executing one line at a time, pausing for operator action between every executable line.
9. The Command Scripter GUI Interface shall have a set of mode control buttons for controlling the execution of the script file. These Mode Control Buttons are:
 - 9.1 **START** Button - starts the execution of the script file.

- 9.2 **STOP** Button - temporarily stops execution of the script file.
- 9.3 **CONTINUE** Button - when a script file has been stopped during execution, or in single step mode, will proceed to execute the next executable line if one exists.
- 9.4 **FAST** Button - changes the current mode of execution to fast.
- 9.5 **SLOW** Button - changes the current mode of execution to slow.
- 9.6 **SINGLE STEP** Button - changes the current mode of execution to single step.
- 9.7 **SET BREAK POINT** Button - toggles a break point for a given executable line which will temporarily stop execution at that line when the script file is run.
- 10. The Command Scripter shall temporarily stop execution of a script file upon error in command execution and wait for action from operator.

5.2.3 Command Scripter Performance Requirements

Command Scripter performance requirements are not defined.

5.2.4 Command Scripter Interfaces Data Flow Diagram

Figure 2-1 shows the data flow for the interface to the Command Processor. The command is entered from the keyboard to the Command Processor GUI. The command string is then passed to the parser/builder which returns a Command object containing an error code indicating if the command was syntactically correct. If applicable to the particular command, a Two Step Window is displayed requesting confirmation before the command is sent to Command Management on the CCWS.

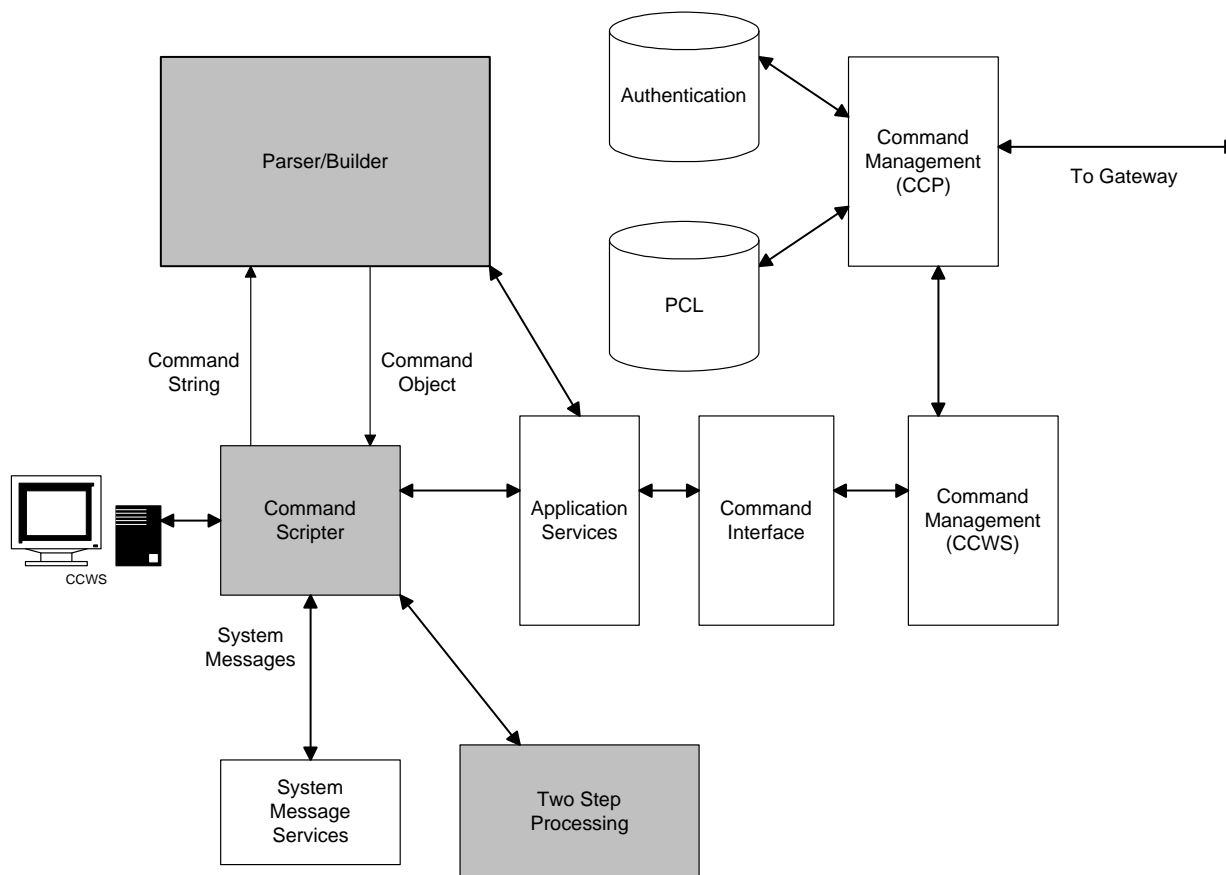


Figure 5-1 Command Scripter Interfaces Data Flow Diagram

5.3 COMMAND SCRIPTER DESIGN SPECIFICATION

5.3.1 Command Scripter Class Diagrams

The Command Scripter Class Diagrams are shown in Figure 5-3. The Command Scripter class is the central class of Command Scripter. It contains the methods for controlling where a command is sent in various stages of processing and how the Command Scripter operates during different modes. The Parser class does all processing of the command string, including syntax and error checking. The TwoStepWindow class is created to arm a two step command. It displays a GUI which lists the command and it's options and gives the user the choice to cancel or send the command. The Command class is used to pass back information on a parsed command to the Command Processor. It includes a status code and an instance of a commanding object.

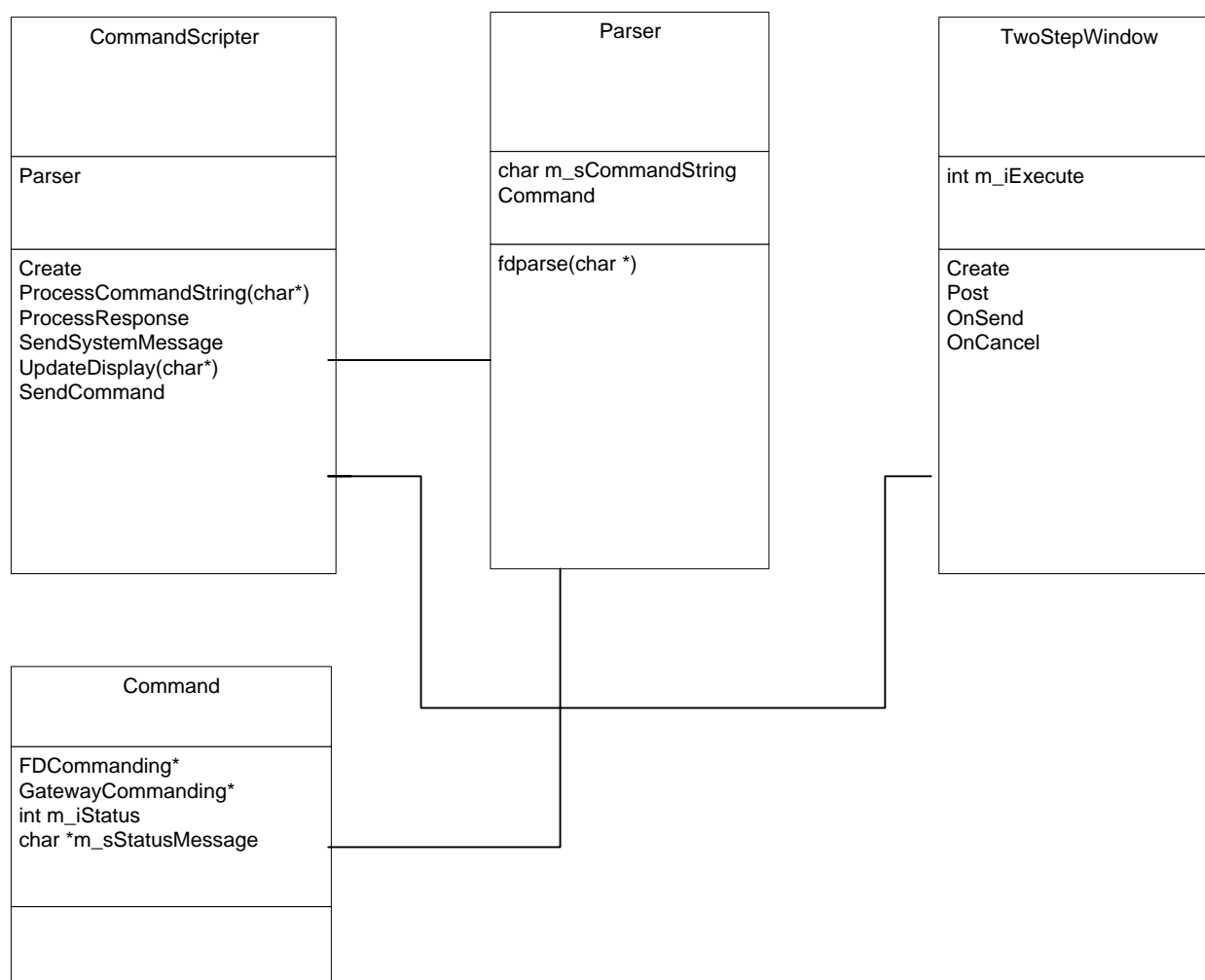


Figure 5-3 Command Scripter Class Diagrams

5.3.2 Command Scripter External Interfaces

5.3.2.1 Command Scripter Message Formats

See Command Processor section 2.3.2.1.

5.3.2.2 Command Scripter Display Formats

The Command Scripter display is shown in Figure 5-4.

A menu bar at the top provides various options for editing script files. The File menu gives options to open, close, and save scripts. These operations are performed through standard file manager dialog boxes. Any errors in scripting syntax (not command syntax) will be checked for when opening a script file. The Edit menu gives options to copy, cut, and paste lines within a script.

The list box shows the contents of a script file. It allows a command line in the script to be selected and edited. It also supports standard cut, copy, and paste functions.

A *Start* button is provided to begin executing a script. When a script is currently executing, this button changes to display *Stop*, which will suspend execution of the script.

A *Continue* button is provided to execute the next executable line of a script when a script is stopped or when executing in single step mode.

A *Break* button is provided to set a breakpoint at a selected line in a script.

The Current Mode text area displays the mode that the script is currently executing in: Fast, Slow, or single Step.

The Mode Override radio buttons provide the capability to override the currently executing mode. They can be used to set the mode in either Fast, Slow, Single Step, or No Override.

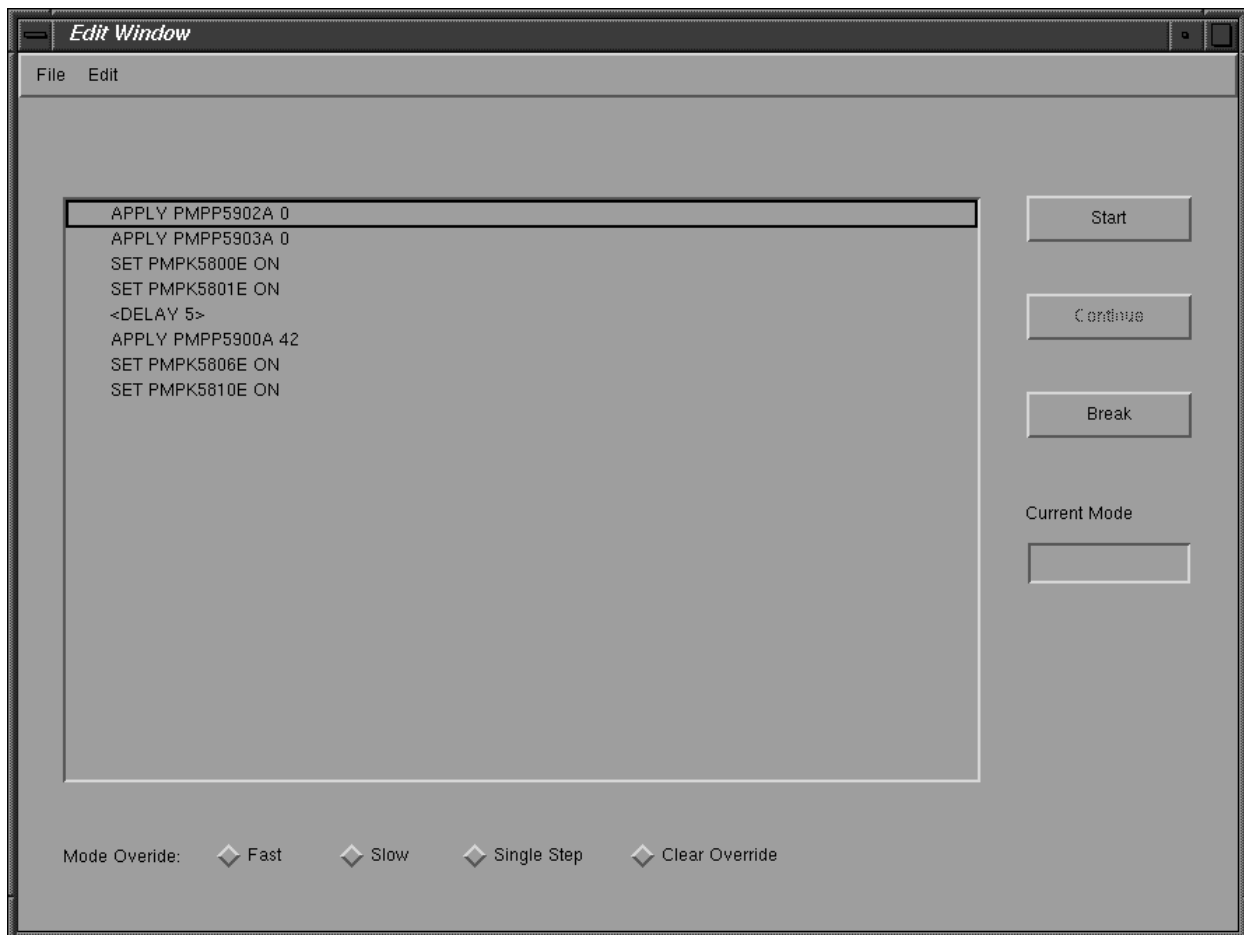


Figure 5-4

5.3.3 Command Scripter Input Formats

Command Scripter Input Formats will follow the syntax rules given in Appendix A CLCS Commands and the Scripter commands given in section.5.2.2.9.

5.3.4 Command Scripter Recorded Data

N/A

5.3.5 Command Scripter Printer Formats

N/A

5.3.6 Command Scripter Interprocess Communications

System Services will be used to support interprocess communications between Command Scripter and Command Management.

5.3.7 Command Scripter External Interface Calls

Command Scripter will make external calls to System Messages and Application Services. System Messages has been addressed in section 5.3.3. Application Services will be used to extract data for validation.

5.3.8 Command Scripter Table Formats

N/A

5.3.9 Command Scripter Test Plan

- 1 Verify the Command Scripter accepts command syntax as specified in Appendix A.
- 2 Verify the Command Scripter accepts scripter syntax as specified in section 5.2.2.9.
- 3 Verify the Command Scripter returns an appropriate error code on entering erroneous commands.
- 4 Verify the Command Scripter issues command requests.
- 5 Verify the Command Scripter supports two-step Command Requests (arm first, then execute).
- 6 Verify the Command Scripter supports imbedded commands as given in section 5.2.2.8.

6. AUTHENTICATION SERVICE CSC

6.1 AUTHENTICATION SERVICE INTRODUCTION

Thor Command Support provides the initial authentication capabilities. The requirements provided below will be augmented with prototype authentication system support code. As the overall authentication design progresses, the Authentication Service requirements may be affected.

The purpose of Authentication is to assure that commands are issued only from authorized sources. An authorized source is either an authorized CCWS or an authorized EIM/EIC. For example, if a command is issued from a viewer on a CCWS, then the CCWS must be authorized to issue that particular command. If an EIM/EIC issues a command, then that EIM/EIC must be authorized to issue the command. Notice that the first case associates authority with a physical device and the second case associates authority with an application.

Authentication Service is invoked by a client application and the resulting disposition is returned to the client. The client must assure that the command does not progress in the system if authority is denied. The client applications are Command Management and the CORBA Filter. Furthermore, a system message will report any authentication disapproval.

There are several types of commands which must be authenticated. They include FD commands, commands associated with FDs, commands associated with Gateways, and commands invoked via an ORB (Object Request Broker). Furthermore, there may be commands which do not fit one of these categories.

6.1.1 Authentication Service Overview

The interface to Authentication is bi-directional as shown in Figure 6-1. The input shall consist of the command and the necessary information to authenticate the command. For the purposes of this discussion, the input information is collectively referred to as the command information. The output consist of a disposition of the authentication request. Additionally, a system message is sent for negative authentication responses.

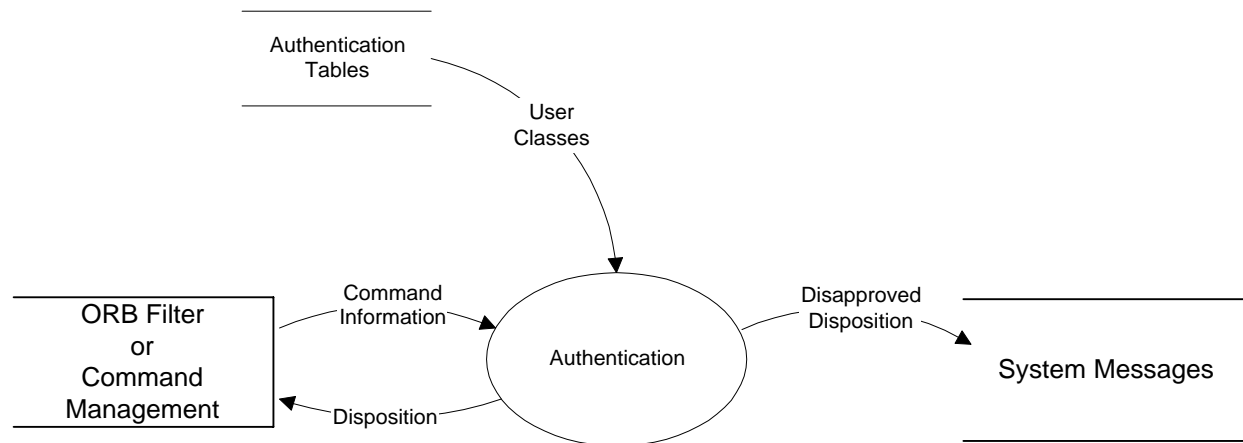


Figure 6-1 Authentication Service Macroscopic Data Flow

6.1.2 Authentication Service Operational Description

6.2 AUTHENTICATION SERVICE SPECIFICATIONS

6.2.1 Authentication Service Ground Rules

The ground rules define the basis for developing the Authentication Service. The specific rules follow:

- For CCWS invocations of Authentication Service, the sub-system-identifierRTPS packet shall be passed via the interface.
- For EIM/EIC invocations of Authentication Service, the application-name source CPU, source application, and destination application shall be passed via the interface.
- The client of the Authentication Service must properly handle negative responses. It is very important to note that it is not Authentication's responsibility to stop the processing of a command. Therefore, the client must stop the progression of the command based on a negative authentication response.
- CORBA Facility must be available.

6.2.2 Authentication Service Functional Requirements

The interface input/output follows:

1. The input to the Authentication Service shall be the command information. The command information shall not be modified by the Authentication Service.
2. The output of the Authentication Service shall consist of one of the following return responses:
 - A) Command is authorized.
 - B) Command is not a member of the TCID.

- C) Command is not associated with a user class assigned to the CCWS.
 - D) Command is not authorized for the application.
 - E) Erroneous command information.
3. If the command is not authorized, then the Authentication Service shall issue a system message with the following information:
- A) Reason for the disapproval:
 - 1) Command is not a member of the TCID.
 - 2) Command is not associated with a user class assigned to the CCWS.
 - 3) Command is not authorized for the application.
 - 4) Erroneous command information.
 - B) The name of the command.
 - C) For CCWS initiated commands, the name of the initiating sub-system.
 - D) For EIM/EIC initiated commands, the name of the initiating application and initiating sub-system.
4. The interface to the Authentication Service shall support invocation on CCWSs and CCPs.

6.2.3 Authentication Service Performance Requirements

Authentication Service performance requirements are not defined. The Authentication Service shall process an authentication request in a period of time that does not impact the requirements specified in section 3.2.3.

6.2.4 Authentication Service Interfaces Data Flow Diagram

Figure 6-1 shows the data flow for the interface to the Authentication Service. The command information is passed to authentication and the disposition is returned. A system message is only sent if a disapproval occurs.

6.3 AUTHENTICATION SERVICE DESIGN SPECIFICATION

Authentication will provide services via a process running on each CCWS and CCP as shown in Figure 6-2. The process will be started via the same image regardless of the sub-system. Therefore, a CCWS and CCP will have the same resident process. The process will be started by OPSCM when the Test Set is configured. At startup, the process will read the tables necessary to define the User Class relationships. Specifically, the tables involved are the OLDB, Gateway, EIM/EIC, CCWS, and TAS tables. Note that the latter three tables are specific to authentication. The table structure will probably change based on the prototype work conducted for Thor.

The Authentication process will be invoked via a CORBA member function invocation. This is true for invocations from both Command Management and the ORB Filter. Since the client of Authentication (i.e. Command Management, ORB filter) and the Authentication process are on the same machine, network traffic will not be generated due to the CORBA invocation.

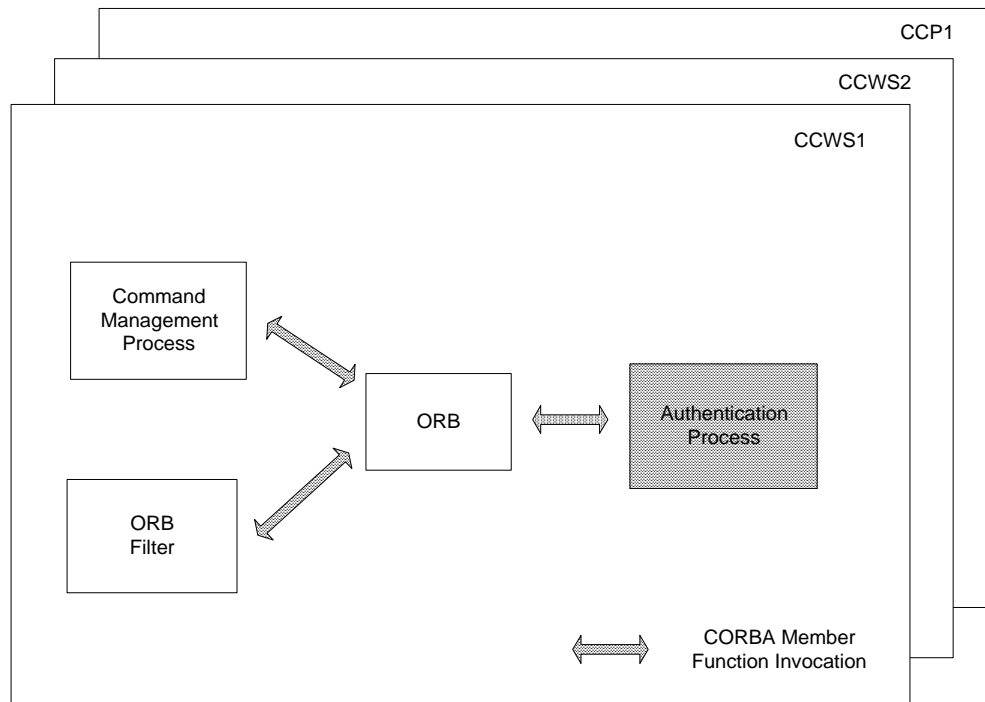


Figure 6-2 Authentication Process

6.3.1 Authentication Service Detailed Design

6.3.1.1 Authentication Service Detailed Data Flow

The detailed Authentication Data Flow is shown in Figure 6-3. Notice that at the top of this diagram is a set of static tables which are used to relate entities to user classes. The two types of entities are commandable items and CCWSs. The source of the data for the commandable items is the OLDB, Gateway Table, and EIM/EIC Table. For example, the OLDB contains fields which relate a FD to a RSYS. The RSYS is treated as a User Class for the Thor prototype. The source of the data for the CCWSs is the CCWS Table. This table relates a particular CCWS to its associated user classes.

As shown in Figure 6-3, the static information is managed via lists which are named after their associated static tables. For example, the EIM_EIC_List is built from the EIM/EIC Table. The list relates the entity to user classes via a mask. The mask is simply a 64 bit field which represents up to 64 user classes. If a bit is set to "1" then the user class corresponding to that bit is associated to the particular entity.

As previously mentioned, Authentication is invoked from Command Management and the ORB Filter. For Command Management invocations, the RTPS packet is passed to Authentication. For ORB Filter invocations, the source CPU, source application, and destination application are passed to authentication.

Invocations via the ORB filter are initiated from two sources. The first source is a user application. The user application invokes an EIM member function. However, before the member function is invoked, the ORB Filter intercepts the invocation and invokes Authentication. An Authentication disapproval stops the EIM member

function invocation. The second source is an EIM invoking a member function of another EIM/EIC. The process relative to the filter is the same as previously described for the viewer invocations of an EIM.

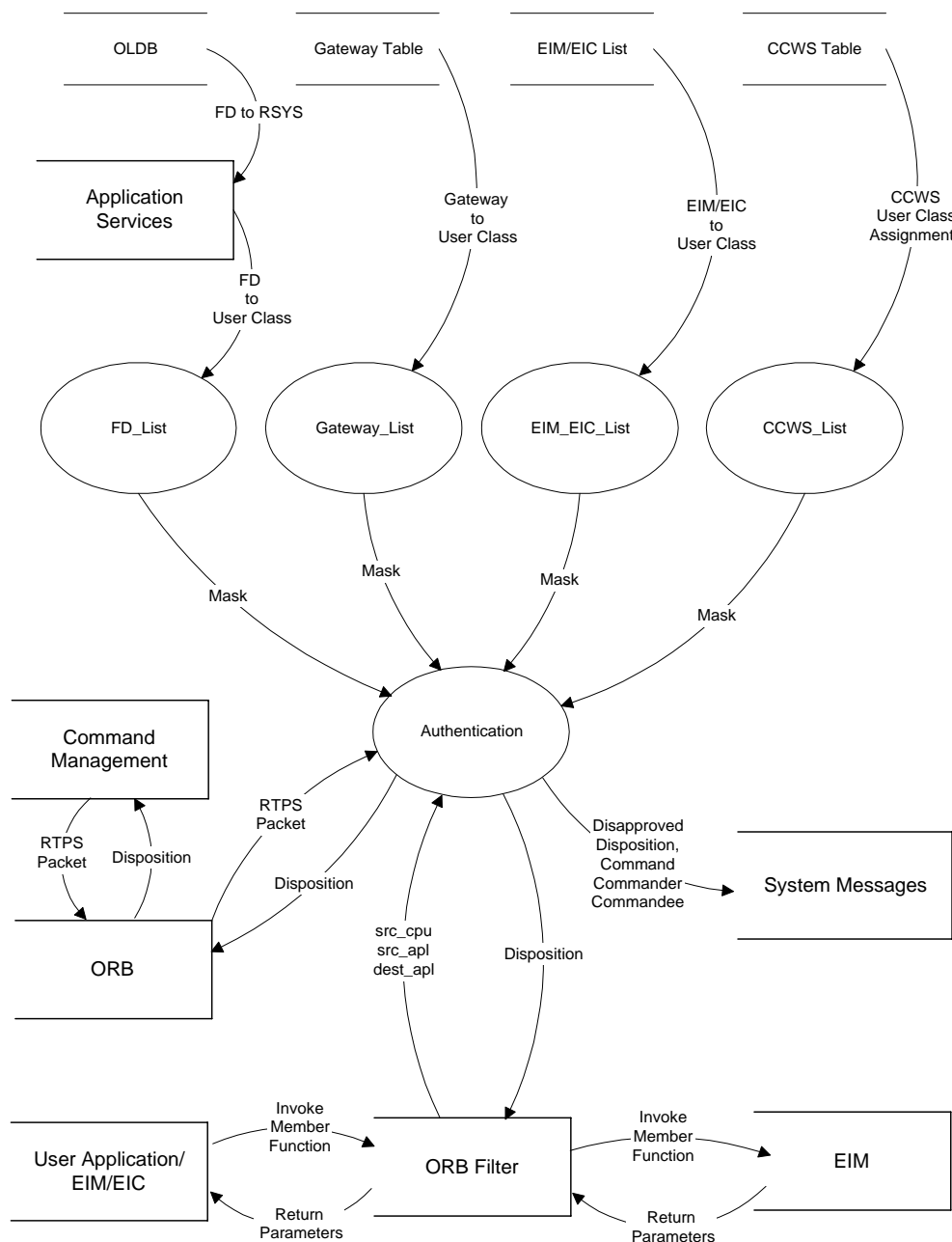


Figure 6-3 Authentication Service Detailed Data Flow

6.3.1.2 Authentication Service Class Diagrams

The Authentication Service Class Diagrams are shown in Figure 6-4. The Authentication class consist of an aggregation of the CCWS_List and Command_List classes. The CCWS_List class reads data from the static CCWS Table in the Detailed Data Flow Diagram (see Figure 6-3). The Command_List class consist of an aggregation of the FD_List, Gateway_List, and EIM_EIC_List classes. These classes also read data from the static files shown in the Detailed Data Flow Diagram.

The list classes maintain an association between an entity and a set of user classes defined by the mask. Therefore, the purpose of the list classes is to return a mask as shown in the Detail Data Flow Diagram and Class Diagram.

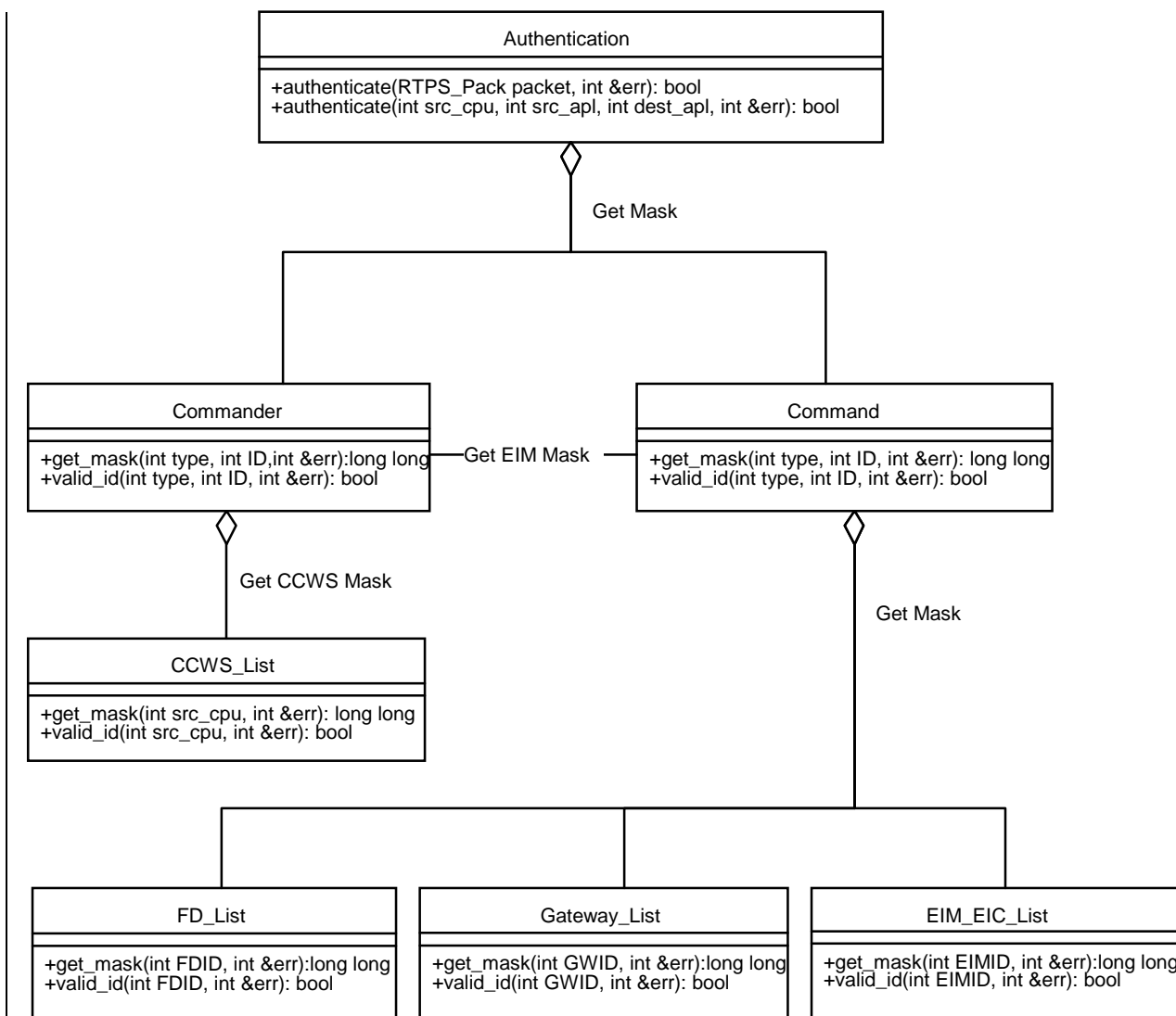


Figure 6-4 Authentication Service Class Diagram

6.3.2 Authentication Service External Interfaces

6.3.2.1 Authentication Service Message Formats

Authentication will send the following messages via System Message Services. Additionally, the system error message number will be returned to the client application via the err parameter.

Message Number	Message Text
1	Command not a member of the TCID.
2	Command not associated to a user class assigned to CCWS.
3	Command not authorized for the application.
4	Erroneous command information.

Message Number: 1

Message Group: CMD

Severity: Warning

Message Text: Command not a member of the TCID. Command: #Insert1 Sub-System: #Insert2
Source Appl: #Insert3

Help Text: The issued command is not a member of the TCID and therefore authentication is denied.

Insert1 = ASCII string representing the command.

Insert2 = ASCII string representing the sub-system (i.e. CCWS1, CCP1).

Insert3 = ASCII string representing the source application.

Message Number: 2

Message Group: CMD

Severity: Warning

Message Text: Command is not associated to a User Class assigned to the CCWS.
Command: #Insert1 Sub-System: #Insert2

Help Text: Authentication compares the list of User Classes assigned to a CCWS and the list of User Classes associated to a Command(i.e. FD, EIM Command). If there is at least one common User Class then authentication is approved. In this case there were no common User Classes.

Insert1 = ASCII string representing the command.

Insert2 = ASCII string representing the sub-system (i.e. CCWS1, CCP1).

Message Number: 3

Message Group: CMD

Severity: Warning

Message Text: The application is not authorized to issue the command. Command: #Insert1
Sub-System: #Insert2 Application: #Insert3

Help Text: Authentication compares the list of User Classes assigned to an application and the list of User Classes associated to a Command(i.e. FD, EIM Command). If there is at least one common User Class then authentication is approved. In this case there were no common User Classes.

Insert1 = ASCII string representing the command.

Insert2 = ASCII string representing the sub-system(i.e. CCWS1, CCP1).

Insert3 = ASCII string representing the source application.

Message Number: 4

Message Group: CMD

Severity: Error

Message Text: Erroneous command. Command: #Insert1 Sub-System: #Insert2 Application: #Insert3

Help Text: Some/all of the input data to Authentication is incomprehensible. This condition is due to either a hardware anomaly or a software bug.

Insert1 = ASCII string representing the command.

Insert2 = ASCII string representing the sub-system(i.e. CCWS1, CCP1).

Insert3 = ASCII string representing the source application.

6.3.2.2 Authentication Service Display Formats

No Authentication displays will be delivered as part of Thor. However, displays will be a constituent of Authentication in post Thor threads.

6.3.2.3 Authentication Service Input Formats

Authentication receives inputs from two sources. The first source is static tables and the second is via parameters passed by member function invocations.

The static tables are in ASCII format and are described in Section 1.3.2.8.

The member function parameters are unique for Command Management and ORB Filter invocations. Command Management invocations pass the RTPS packet as defined by the Real Time Processing System Packet Payload ICD (84K00351). Invocations from the ORB Filter pass the source CPU, source application, and destination application.

6.3.2.4 Authentication Service Recorded Data

N/A

6.3.2.5 Authentication Service Printer Formats

N/A

6.3.2.6 Authentication Service Interprocess Communications

The interprocess communications between Authentication and its clients will be accomplished via CORBA. Messages will be compliant with OMG CORBA 2.0 IDL.

6.3.2.7 Authentication Service External Interface Calls

Authentication will make external calls to System Messages and Application Services. System Messages has been addressed in section 1.3.2.1. Application Services will be used to extract data from the OLDB. This interface is described in the document Application Services Requirements and Design (84K00520).

6.3.2.8 Authentication Service Table Formats

The four static tables which are used to initialize Authentication are all ASCII formats. The OLDB is an existing format which will be accessed via Application Services.

The other three files are defined by Authentication Services and are described as follows. Notice that a particular entity (i.e. gateway, EIM, CCWS) can appear in more than one record. This provides a method of associating more than one User Class to an entity.

Gateway Table file format:

GWID1	User_Class1<Cr>
GWID1	User_Class2<Cr>
GWID2	User_Class2<Cr>

EIM Table file format:

EIMID1	User_Class1<Cr>
EIMID1	User_Class2<Cr>
EIMID2	User_Class2<Cr>

CCWS Table file format:

CCWS1	User_Class1<Cr>
CCWS1	User_Class2<Cr>
CCWS2	User_Class2<Cr>

6.3.3 Authentication Service Test Plan

The following are the required test which validate that Authentication meets the requirements on both the CCWSs and CCPs.

- Via code inspection, verify that the command information passed to Authentication is declared “const”. The “const” declaration assures that the parameters cannot be modified by the member function. The only modified parameter is the “err” parameter which returns the error status.
- Issue the following type of commands from the listed sub-system and verify the expected results. Note that testing for erroneous parameters requires a test tool. Erroneous parameters only occur due to software bugs or hardware failures.

Command	Source	User Class Association	Disposition	System Error Message ¹
FD	CCWS CPROC	Commander Commandee Match	Approved	None
FD	CCWS CPROC	Commandee not in TCID	Disapproved	Error 1
FD	CCWS CPROC	No Commander Commandee Match	Disapproved	Error 2
FD	CCWS CPROC	Erroneous Parameters	Disapproved	Error 4
FD	CCWS Viewer	Commander Commandee Match	Approved	None
FD	CCWS Viewer	Commandee not in TCID	Disapproved	Error 1
FD	CCWS Viewer	No Commander Commandee Match	Disapproved	Error 2
FD	CCWS Viewer	Erroneous Parameters	Disapproved	Error 4
FD	CCP EIM	Commander Commandee Match	Approved	None
FD	CCP EIM	Commandee not in TCID	Disapproved	Error 1
FD	CCP EIM	No Commander Commandee Match	Disapproved	Error 3
FD	CCP EIM	Erroneous Parameters	Disapproved	Error 4
Gateway	CCWS CPROC	Commander Commandee Match	Approved	None
Gateway	CCWS CPROC	Commandee not in TCID	Disapproved	Error 1
Gateway	CCWS CPROC	No Commander Commandee Match	Disapproved	Error 2
Gateway	CCWS CPROC	Erroneous Parameters	Disapproved	Error 4
Gateway	CCWS Viewer	Commander Commandee Match	Approved	None
Gateway	CCWS Viewer	Commandee not in TCID	Disapproved	Error 1
Gateway	CCWS Viewer	No Commander Commandee Match	Disapproved	Error 2
Gateway	CCWS Viewer	Erroneous Parameters	Disapproved	Error 4
Gateway	CCP EIM	Commander Commandee Match	Approved	None
Gateway	CCP EIM	Commandee not in TCID	Disapproved	Error 1
Gateway	CCP EIM	No Commander Commandee Match	Disapproved	Error 3
Gateway	CCP EIM	Erroneous Parameters	Disapproved	Error 4
EIM	CCWS Viewer	Commander Commandee Match	Approved	None
EIM	CCWS Viewer	Commandee not in TCID	Disapproved	Error 1

¹ See Section 6.3.2.1 for error code definitions.

EIM	CCWS Viewer	No Commander Commandee Match	Disapproved	Error 2
EIM	CCWS Viewer	Erroneous Parameters	Disapproved	Error 4
EIM	CCP EIM	Commander Commandee Match	Approved	None
EIM	CCP EIM	Commandee not in TCID	Disapproved	Error 1
EIM	CCP EIM	No Commander Commandee Match	Disapproved	Error 3
EIM	CCP EIM	Erroneous Parameters	Disapproved	Error 4

As part of the prototype work, the performance of the Authentication Service must be quantified. On both a CCWS and CCP the maximum authentication rate must be determined via a test tool. The rate will be expressed in commands authenticated per second.